N.C. State U.
Raleigh, N.C.

TELEROBOTIC CONTROL OF A MOBILE COORDINATED ROBOTIC SERVER

NAG-1-1283-2

ANNUAL TECHNICAL REPORT

Executive Summary

This annual report is comprised primarily of results from the Master's Degree Thesis of Mr. Darrell Gerber, a graduate student supervised by the principal investigator on this project. The goal of this effort is to develop advanced control methods for flexible space manipulator systems. As such, an adaptive fuzzy logic controller has been developed in which model structure as well as parameter constraints are not required for compensation. The work builds upon previous work on fuzzy logic controllers. Fuzzy logic controllers have been growing in importance in the field of automatic feedback control. Hardware controllers using fuzzy logic have become available as an alternative to the traditional PID controllers. Software has also been introduced to aid in the development of fuzzy logic rule-bases. The advantages of using fuzzy logic controllers include the ability to merge the experience and intuition of expert operators into the rule-base and that a model of the system is not required to construct the controller. A drawback of the classical fuzzy logic controller, however, is the many parameters need to be tuned off-line prior to application in the closed-loop. In this report, an adaptive fuzzy logic controller is developed requiring no system model or model structure. The rule-base is defined to approximate a state-feedback controller while a second fuzzy logic algorithm varies, on-line, parameters of the defining controller. Results indicate the approach is viable for on-line adaptive control of systems when the model is too complex or uncertain for application of other more classical control techniques.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER II

CHAPTER III

CHAPTER IV

LIST OF SYMBOLS

ce ..... change in error
ce ..... change in error in position
ce(k) ..... current change in error in position
d ..... displacement of the control plane along the e-axis
D ..... derivative parameter
e ..... error
e ..... error in position
e(k) ..... current error in position
e(k-1) ..... previous error in position
$\dot{e}$ ..... time rate of change of error in position
$\overline{E}$ ..... weighted running sum of the error in position
$\overline{E}_o$ ..... $\overline{E}$ from previous time step
g ..... acceleration due to gravity (9.81 m/s^2)
k ..... current time step
L ..... length of the vertical pendulum arm
M ..... mass matrix for DR106
n ..... number of time steps taken
O(k) ..... output to the system
P ..... proportional parameter
QE(k) ..... quantized error in position
QECA(k) ..... quantized change in error angle
QO ..... quantized output
QO(k) ..... current quantized output
r(k) ..... current reference input
$\tilde{R}$ ..... vector of non-linear terms for DR106
$\overline{sse}$ ..... approximate steady-state error
$\tilde{T}$ ..... torque vector for DR106
u ..... PD controller output
u ..... results from the antecedent blocks
u_i ..... result from the i^{th} antecedent block
U ..... center values of the consequence block membership functions
U_i ..... center value of the membership function of the i^{th} consequence block
$\overline{(V^2)}$ ..... weighted running sum of the square of the error in velocity
$\overline{(V^2)}_o$ ..... $\overline{(V^2)}$ from the previous time step
y(k) ..... system response

Δd ..... change in d
ϕ ..... inclination of the control plane from the e-$\dot{e}$ plane
γ ..... velocity penalty factor
λ ..... forgetting factor
$\mu(\_)$ ..... membership value to fuzzy set (_)
θ ..... rotation of the control plane about the u-axis
θ ..... angular position of the vertical pendulum

$\tilde{\theta}$ ..... position vector for DR106
θ_n ..... angle of joint n of DR106
τ ..... control forces to the vertical pendulum
τ_n ..... torque applied to joint n of DR106

LIST OF ABBREVIATIONS

MRAC ..... Model Reference Adaptive Control
PD ..... Proportional-Derivative
PID ..... Proportional-Integral-Derivative

CHAPTER I: INTRODUCTION

Many of the current and future control applications are too complex for traditional controllers due to the presence of non-linearities, varying or uncertain parameters, and/or time delays. These applications require the use of a robust adaptive controller. Many adaptive controllers (MRAC, Self-Tuning Adaptive Controllers, joint space control [1], and global linearization [2], for example) have been offered; however, developing a model of sufficient accuracy may be impossible or too difficult to be practical.

Fuzzy logic controllers have been shown to be an acceptable alternative to model based controllers [3,4]. Fuzzy PID control has been applied to several process control and automotive systems [5-7] in which the time constants were somewhat large. The use of fuzzy logic to control robotic systems has yielded some success [8-11] although issues such as time delays and initial conditions sometimes limit the application of these algorithms.

The rule-base of a standard fuzzy logic controller is developed from linguistic rules stating the reaction of an experienced operator to various situations. An alternate approach is to choose the rule-base to approximate the behavior of another controller. The advantage of this is that the versatility of the fuzzy logic controller and the known characteristics of the approximated controller are both retained.

The flexibility of the fuzzy logic controller is counteracted, however, by the increased number of parameters that need to be tuned. Various adaptive fuzzy logic controllers have been offered as a solution to this problem. Many of them, however, circumvent the advantages of a fuzzy logic controller by requiring that a model of the system be known. This thesis develops an adaptive fuzzy logic controller that is able to handle the uncertainties in complex systems without requiring a system model.

As an extension of previous work [12], the approach uses a PD structure to form the rule-base. The parameters of the rule-base are varied on-line by an adaptation routine

utilizing another fuzzy logic controller. The method is applied to a vertical pendulum as well as a robotic system to illustrate its applicability.

Chapter II gives a brief overview of fuzzy logic controllers, while Chapter III develops the tuning algorithm. In Chapter IV the adaptive fuzzy logic controller is applied to several complex systems. Results indicate the technique is a robust alternative to traditional solutions. Finally, Chapter IV provides some concluding remarks as well as areas of potential future work.

CHAPTER II: NON-ADAPTIVE FUZZY LOGIC CONTROLLER

Before an adaptive fuzzy logic controller is attempted, one should have a basic understanding of how a non-adaptive fuzzy logic controller works. The structure of a fuzzy logic controller can be represented as in Figure 1.

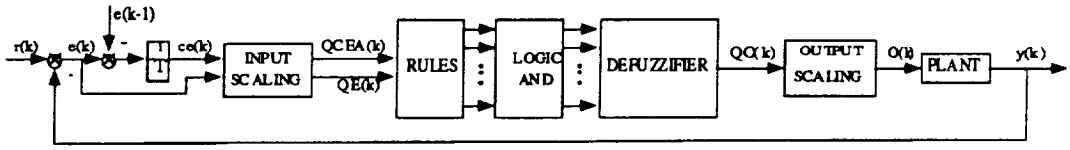


Figure 1: Non-adaptive fuzzy logic controller block diagram

From this figure, the important parts of a fuzzy logic controller can be seen; these are the

A. Rule-base

B. Defuzzifier, and

C. Input and Output Scaling

Each of these components will now be dealt with.

A: THE RULE-BASE

The rule-base is the heart of every fuzzy logic controller. As can be inferred, the fuzzy logic controller uses a number of rules to define the relationship between the input and the output of the controller. What differentiates a fuzzy logic controller from other rule-based controllers is that it uses methods of fuzzy implication and compositional rules of inference.

The rules in the fuzzy logic controller are based on linguistic rules of the form

IF *one condition* **AND** *another condition* **THEN** *do something*

At the start these rules vary greatly depending on the application and the individual interpretation of the situation being described by the rule. For instance, if describing the manual control of the temperature in a room, one rule could be

IF *the temperature is much too hot* **AND** *the temperature is dropping fairly fast* **THEN** *turn the furnace up a little bit*

Of course this is only one way of stating this particular situation and action. To eliminate the inconsistency that is inherent with this type of formulation a standard vocabulary is adopted. This new vocabulary consists of words such as LARGE, MEDIUM, SMALL and POSITIVE, NEGATIVE, ZERO. Stating the previous example using this new vocabulary changes it to

IF *error in temperature is large positive* **AND** *change of error in temperature is medium negative* **THEN** *change furnace small positive*

Even though this is a more general and tractable statement it is still no more
or concrete. *Error in temperature*, *change of error in temperature*, and *change*
' exact measurements but what are *large positive*, *medium negative*, and *small*
'w are they all related?

'c is uniquely appropriate here because people do not usually think of
"large" or "not large" but rather think "how large is it?" This

illustrates the difference between what is usually thought of as logic (Boolean logic) and fuzzy logic. Boolean logic deals with whether something is or is not a member while fuzzy logic deals with varying degrees of membership. It should also be mentioned that, contrary to what is often thought, fuzzy logic has nothing to do with probability. Fuzzy logic does not consider the probability of membership – only the degree of membership.

The degree of membership is determined by a membership function. This is the connection between the "fuzziness" of the real world and the exactness of mathematics. Membership functions can be of any shape but they must only have values between 0 and 1; that is, between *no membership* and *complete membership*. Most membership functions are symmetric and simple. Several common shapes are shown in Figure 2. Notice that Boolean logic can be considered a subset of fuzzy logic as the last membership function shows.

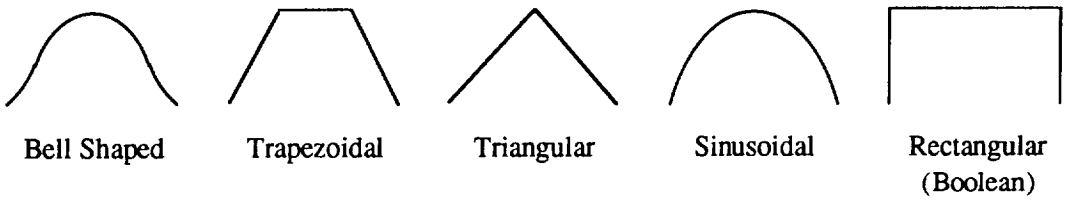


Figure 2: Common membership functions

The shape of the membership function has been shown to be nearly arbitrary [11]. Therefore, the choice can be based on such things as ease of calculation, appropriateness to the application, and personal preference. The fuzzy logic controllers developed here use a sinusoidal membership function. This is because it is smooth, symmetrical, and based on a simple mathematical function.

The application of the membership function requires the introduction of the term *universe of discourse*. The universe of discourse is the minimum region over which the variable of interest is expected to exist. The membership function will be defined within this same region. To get a better understanding, let us look at a simple example. Consider a universe of discourse of the possible heights of full grown males. Upon this, define a triangular membership function TALL centered on 6'2" and intersecting the axis at 5'11" and 6'6". This is shown graphically in Figure 3.

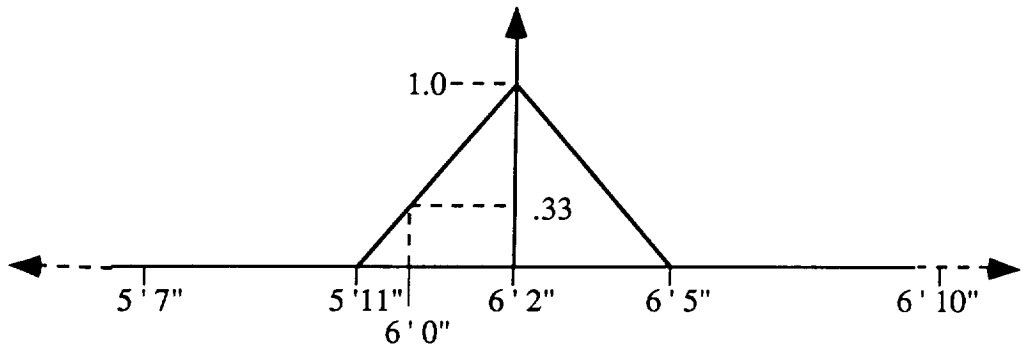


Figure 3: Example finding membership value of the membership function TALL

On the basis of on this we want to determine how TALL a guy 6'0" is. It is easy to see that he is .33 (or 33%) TALL.

Now that the rule-base, membership function and universe of discourse have been introduced their relationship needs to be determined. To facilitate a more organized discussion, the structure of the rules needs to be better defined. Therefore, consider rules of the form

IF *antecedent block* **THEN** *consequence block*

Up to this point, only individual rules dealing with a single situation have been considered. However, an effective controller must be able to handle many distinct situations. This requires that several different rules be developed. The accumulation of these rules is the rule-base. The first step in developing the rule-base is to determine how many elements will be in the antecedent and consequence blocks. These correspond to the number of inputs and outputs, respectively. The second step is to determine how many distinct membership functions will be defined for each element of the antecedent and consequence blocks. For the rest of this development we will use a combination of a two input, one output rule-base with seven distinct membership functions (LARGE NEGATIVE, MEDIUM NEGATIVE, SMALL NEGATIVE, ZERO, SMALL POSITIVE, MEDIUM POSITIVE, and LARGE POSITIVE) for each. The membership functions will be centered such that no more than two will have non-zero values at any time. A sample rule-base is shown in Figure 4.

IF	*error is LN*	**AND**	*change in error is LN*	**THEN**	*LP*
IF	*error is LN*	**AND**	*change in error is MN*	**THEN**	*LP*
IF	*error is LN*	**AND**	*change in error is SN*	**THEN**	*MP*
	⋮		⋮		⋮
IF	*error is LP*	**AND**	*change in error is LP*	**THEN**	*LN*

Figure 4: Sample rule-base

It should be obvious that the largest possible number of distinct rules for this rule-base is 49 $\left[(7 \text{ membership functions})^{(2 \text{ inputs})}\right]$. When the maximum number of rules is defined the rule-base is fully populated. A fully populated rule-base is usually used; but

it could happen that some of the situations can never possibly occur. In such cases the corresponding rules may be left undefined. An example of this situation is when a robotic arm is prevented from moving above a certain velocity near a delicate instrument.

Figure 5 shows another way of representing the rule-base in which the membership functions are represented graphically. From this it can be seen that more than one rule may apply at any moment. However, it has been shown [15] that for this particular layout of rules no more than four rules will be in effect at any one moment.

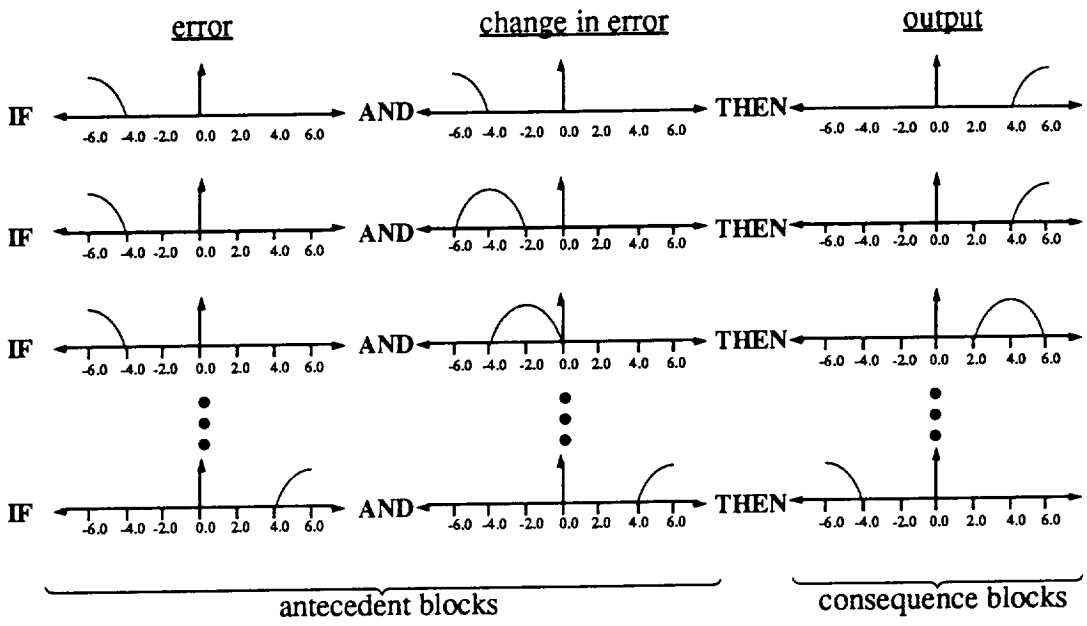


Figure 5: Graphical representation of the sample rule-base

The first step in determining the output is the resolution of the antecedent block using the Logic AND operator. There are various possible definitions of Logic AND [14], a few of which are

i) $\text{AND}(\mu(A),\mu(B)) = \min(\mu(A),\mu(B))$

ii) $\text{AND}(\mu(A),\mu(B)) = \max(0,\mu(A)+\mu(B)-1)$

iii) $\text{AND}(\mu(A),\mu(B)) = \max(1-\mu(A),\mu(B))$

iv) $\text{AND}(\mu(A),\mu(B)) = \min(1,1-\mu(A)+\mu(B))$

The result of each of these is shown in Figure 6 for a given $\mu(A)$ and a varying $\mu(B)$.

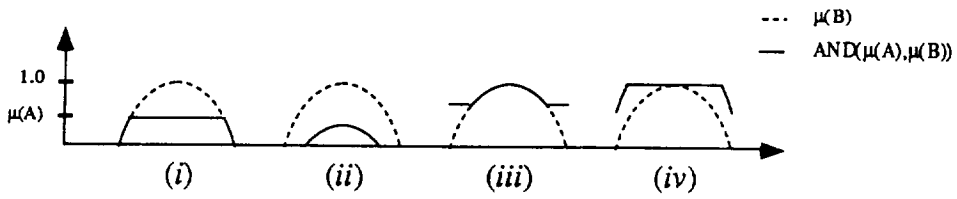


Figure 6: Logic AND

Definition (*i*) will be used throughout this paper. To illustrate how this is evaluated, consider the rule-base in Figure 5 and a situation where error is -6.0 and the change in error is -5.3. Figure 7 shows the evaluation of the antecedent blocks for these inputs.

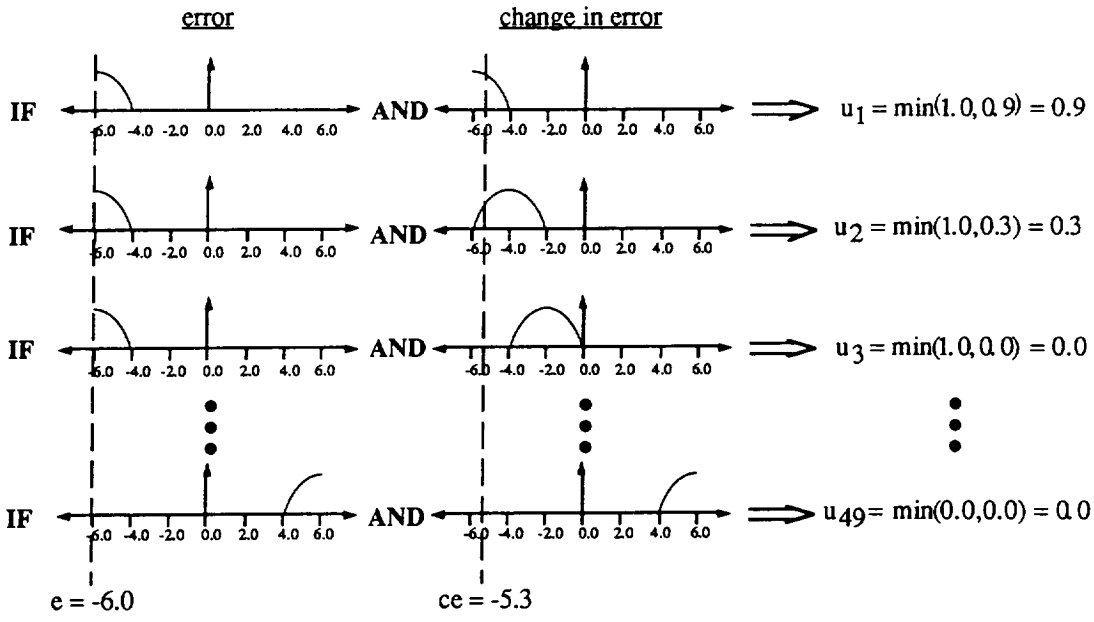


Figure 7: Evaluation of an example antecedent block

B: DEFUZZIFIER

Now that the result of the antecedent blocks has been found, the output from the controller needs to be calculated. Various methods have been proposed [17]. One of these is the *maximum of grade of membership*. In this method

$$QO = \max\{u\}$$

Unfortunately, from nonlinear control system theory, we know this method suffers from poor precision (from dead zone) or unavoidable oscillation (from lack of a dead zone) [18].

Another defuzzifier is the *center of gravity* method given by

$$QO = \frac{\int uUdU}{\int udU}$$

This method gives a better control performance, but the calculation can become quite burdensome due to the iterative algorithms often needed to evaluate the integrals. This leads to the *approximate center of gravity* method.

$$QO = \frac{\sum_{i=1}^{N}(u_i \times U_i)}{\sum_{i=1}^{N} u_i}$$

Due to ease of calculation and superior performance, the approximate center of gravity defuzzifier will be used from here on.

As can be seen it is unnecessary to define the membership functions for the consequence blocks since only the position of the center is needed. This is because the membership functions are necessary only to "fuzzify" a value. Since the results of the antecedent blocks are already fuzzy all that needs to be done is to defuzzify them.

C: INPUT AND OUTPUT SCALING

The universe of discourses used for all the antecedent and consequence blocks shown in Figure 5 range from +6 to -6. It should be obvious that this may not be the most appropriate choice for all possible inputs and outputs. However, to improve the portability of the rule-base it is desirable not to change the universe of discourse for each

new system. A compromise is found by applying a scaling factor to each element depending on the maximum expected values and the size of the generalized universe of discourse.

In applications where the inputs are read through transducers and the outputs go through actuators the scaling factors are dependent on the physical limitations of the components. Another case when the maximum expected value is easily determined is when the inputs and outputs must pass through an analog/digital interface. In this instance the range is dependent on the voltage range used and the number of digital bits used.

Usually the scaling factors are linear relationships; but, this is not necessary. A situation in which a non-linear relationship may be desirable is when the expected values approach $\pm\infty$. This often happens when, as is the case here, the change in error is calculated rather than read from a sensor. A unique solution to this problem was offered by Stanley [12]. He transformed the range of values from $\pm\infty$ to ± 90 by taking the arc tangent of the change in error. The scaling factors to be used in the fuzzy logic controller presented in this thesis are

$$QE = \left(\frac{6}{\text{maximum error}}\right) \bullet (\text{error})$$

$$QCEA = \left(\frac{6}{\text{maximum change in error angle}}\right) \bullet \tan^{-1}\left(\frac{\text{change in error}}{T}\right)$$

$$O = \left(\frac{\text{maximum torque}}{6}\right) \bullet (QO)$$

CHAPTER III: TUNING A FUZZY LOGIC CONTROLLER

Controllers of all varieties need to be tuned in some manner. Fuzzy logic controllers are no different. Unfortunately, the flexibility that is gained by using a fuzzy logic controller is accompanied by an increased number of parameters that need to be tuned. The scaling factors for each input and output can be varied, the shape of the membership functions can be changed, and the rule-base itself may be altered.

There are two ways the scaling factors can be tuned. The first is to change the function itself. A variety of linear or non-linear functions can be used. The other way of tuning the scaling factors is to change the limiting values. This is the most common scheme for auto-tuning and adaptive fuzzy logic controllers [14] since there are fewer possible variations and the effects are more obvious – a small range will give a quicker controller while a larger range gives a more sluggish yet robust controller. The scaling factors are usually tuned off-line depending on the particular system. However, if the plant undergoes known, predictable variations the scaling factors may also be tuned on-line using a gain scheduling routine.

Varying the shape of the membership function, as already mentioned, has little effect on the performance of the controller. However, if one is concerned with the shape of the control surface this is an important factor.

The tuning method used by the adaptation routine developed in this thesis is to vary the rule-base. One reason for this choice is the flexibility in the possible types of variations. Another reason is that while the scaling factors define the fuzzy logic controller's interaction with the rest of the system the rule-base defines the character of the controller. Varying the rule-base is particularly suited to on-line adaptation routines because the performance characteristics of the system are directly related to the rule-base. This will become more evident as the adaptation routine is developed.

A: CHOOSING THE RULE-BASE

When the basic fuzzy logic controller was developed it was stated that the fuzzy rules were based upon linguistic rules derived from human responses to particular situations. There is another possible method for choosing the rule-base. The rule-base can be chosen such that the fuzzy logic controller approximates the performance of another type of controller. The advantage of this method is that controllers whose characteristics can be found analytically (optimal controllers and pole-placement controllers, for example) can then be approximated by the flexible and versatile fuzzy logic controller.

An easy way to visualize this method of choosing a rule-base is with a control surface. The control surface of any controller is found by plotting the outputs versus the inputs. Since a two input/one output configuration has a three-dimensional control surface (which can be easily drawn) this structure will be considered from this point on.

One of the simplest controllers fitting this constraint is the PD controller. Consider the PD controller defined by Eq. (1) and the resulting control surface shown in Figure 8. As can be seen the control surface is simply a plane.

$$u = Pe + D\dot{e} \qquad \text{Eq. (1)}$$

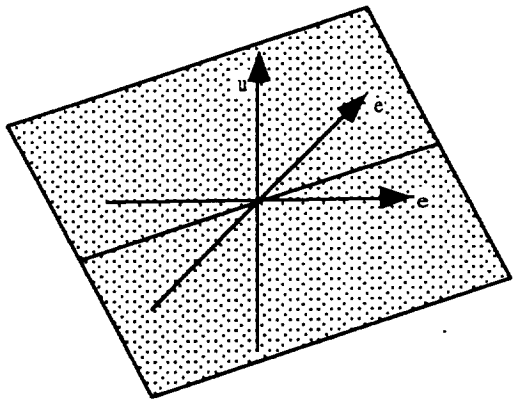


Figure 8: PD control surface

Chen and Jang [15] developed an algorithm to imitate a state feedback controller by correctly choosing the rules in a fuzzy logic rule-base. What their method effectively does is to discretize the input ranges of the state-feedback controller and then define the rules of the fuzzy logic controller to exactly match the value of the state-feedback control surface at these points. The values in between are approximated by the fuzzy logic AND and the approximate center of gravity methods already discussed.

When this technique is used for the PD controller in Eq. (1) we get the fuzzy logic controller having the control surface shown in Figure 9. Two things should be mentioned before continuing. The first is that this controller corresponds to a state-space controller having a 2x1 input vector and a scalar output. The controller also has a planar relationship between input and output. Both of these characteristics were chosen for simplicity of design not because of any restrictions inherent in the methodology. Another important observation is that as more rules are used the fuzzy logic controller will more

closely approximate the state feedback controller. The importance of this depends on the particular application.

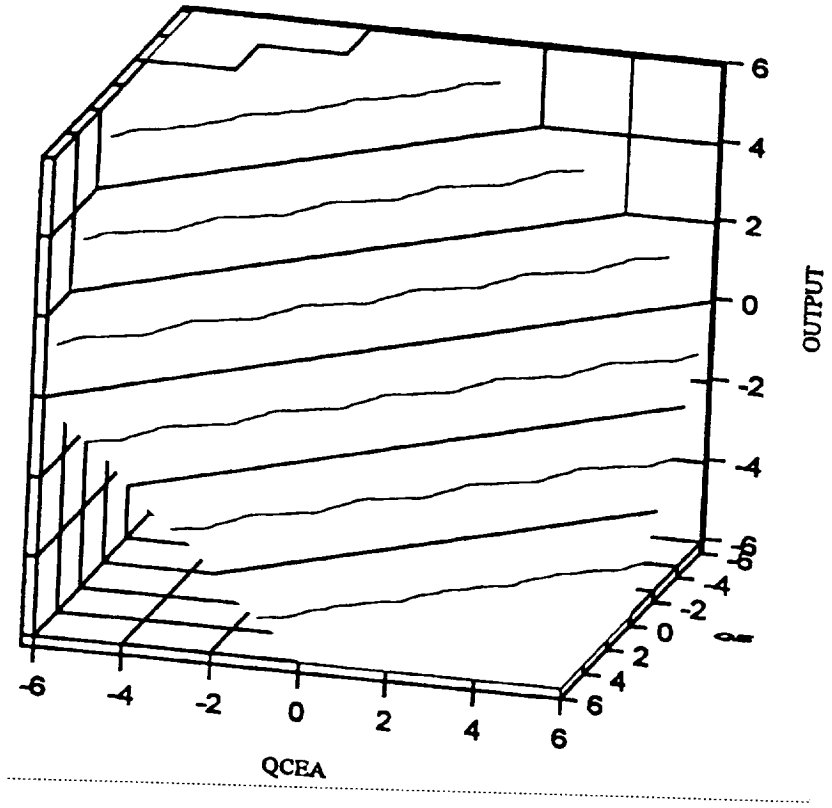


Figure 9: Control surface of a fuzzy logic controller approximating a PD controller

The algorithm developed by Chen and Jang [15] selects the rule-base external to the actual use of the controller. However, there is no reason that a similar function can not be performed while the controller is in operation. This is an important step to the research presented here because it allows the rule-base to be adapted by varying the state-feedback controller that it approximates.

The rule-base is defined by first choosing the desired state-feedback controller. The position of each rule is given by coordinates in the phase plane. The equation defining the state-feedback controller and the coordinates for each rule are then used to define the consequence block of that rule.

As already mentioned, the state feedback controller used is of the same dimensions as Eq. (1). However, rather that define the controller using the P and D variables, it is defined by the rotation (θ) and the inclination (ϕ) of the control surface. (Eq. (2),Figure 10) Another degree of freedom is added by allowing the surface to be displaced along the error axis (d). The importance of this becomes evident later. Selection of these control parameters transforms the controller definition from a mathematical equation into a more visualizable, graphical definition.

$$u = -\frac{\tan\phi}{\tan\theta}e - \tan\phi\dot{e} + d\frac{\tan\phi}{\tan\theta} \tag{2}$$

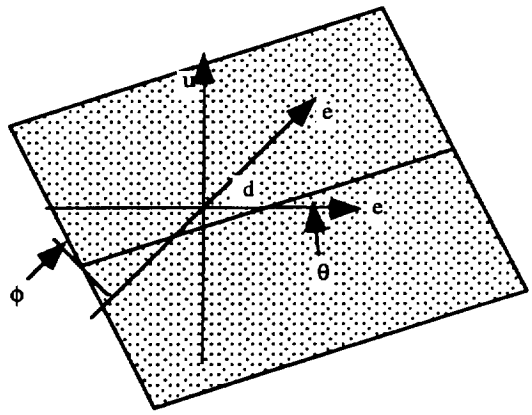


Figure 10: Variables defining the control plane orientation

B: ADAPTATION ROUTINE

Since the rule-base has been defined based on a state feedback controller it should be obvious that the adaptation methods used for state-space controllers can also be used for the fuzzy logic controller. This is what has been done several times in the literature [16-18]. However, direct application of this philosophy seems to defeat one of the primary reasons for using a fuzzy logic controller. One of the driving advantages of the fuzzy logic controller is that it requires no prior knowledge of the plant; but, the adaptation routines proposed for a fuzzy logic controller thus far require that a model of the plant be known. Incidentally, the initial conditions of the adaptive fuzzy logic controller could be found using various techniques (Ziegler-Nichols and Kalman, for example). This may not be necessary, though, because the effects of mistuning can be compensated for both on- and off-line.

One obvious alternative to this philosophy is to use another fuzzy logic controller in the adaptation routine. Hence, the output will be checked using another rule-base to determine the necessary variation in the main controller rule-base. To develop the adaptation rule-base, though, one must understand the effects of varying the different parameters in Eq. (2).

A simple vertical pendulum, shown in Figure 11, is used to better understand the effects of varying the parameters in Eq. (2). The system consists of a point mass at the end of a massless rod. They are attached to a stationary, frictionless, pinned joint. However, to demonstrate the flexibility of the fuzzy logic controller, the usual small angle assumption will not be made.

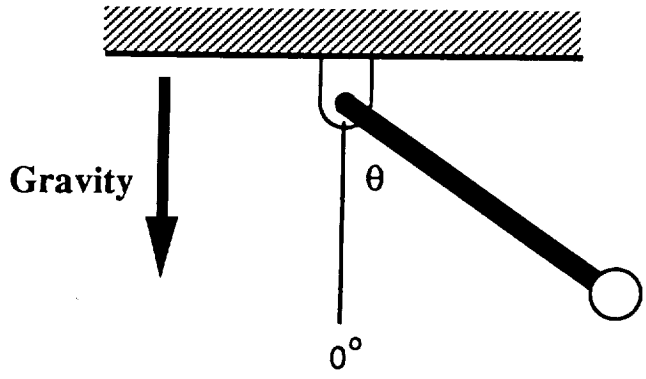


Figure 11: Vertical Pendulum

The dynamics associated with this systems are

$$\ddot{\theta} + \frac{g}{L}\sin\theta = \tau$$

The desired trajectory is a step response of 1.0_{rad}. To make the response a little more interesting, atypical initial conditions are used: $\theta(0) = 2.0_{rad}, \dot{\theta}(0) = -1.0_{rad/s}$.

From the responses shown in Figure 12 we see the primary effects of varying each of the parameters in Eq. (2): d changes the steady-state error, θ changes the damping rate, and ϕ changes the frequency. This is only the most significant variant of each performance characteristic, though. This can be seen from Eq. (2) where the third term is a function of d, θ and ϕ, for example.

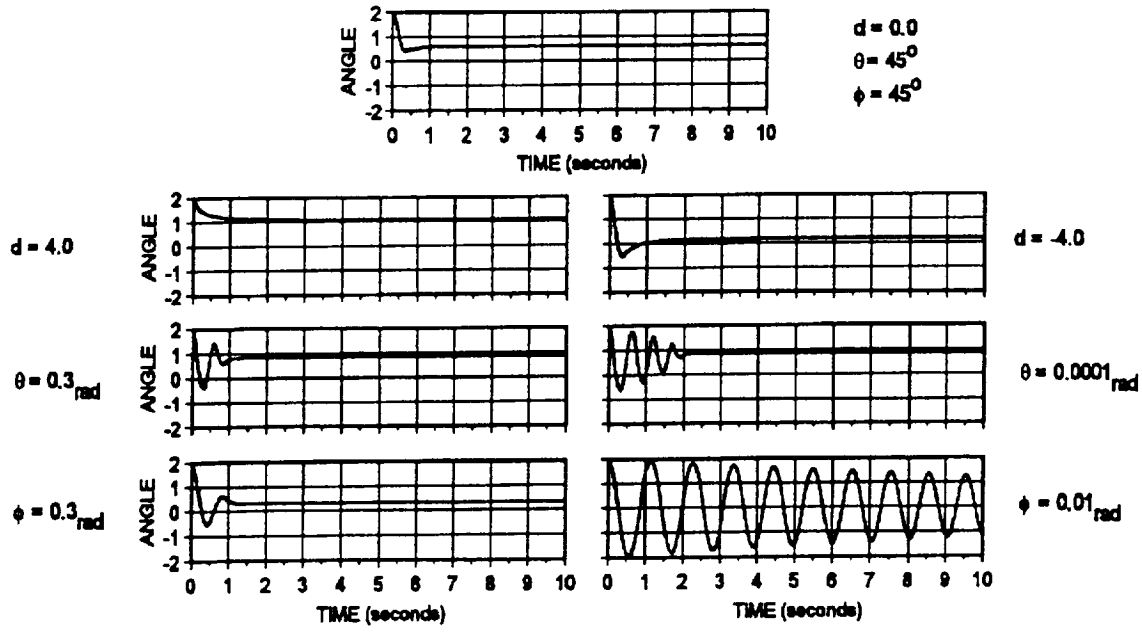


Figure 12: The effects on the vertical pendulum of varying d, θ, and ϕ

It was promised that the importance of adding the d parameter would become evident – this is the time to illustrate this. All that is done by adding the third parameter, d, is to give the controller a constant bias (evident in the changing steady-state error). However, if this parameter is varied adaptively, the effect is the same as having an integrator. In effect, we have designed a pseudo-fuzzy PID controller. For the remainder of this thesis, only the variation of d will be investigated. The adaptation of the other two parameters is the subject of on-going research.

C: PERFORMANCE CHARACTERISTICS

It is now necessary to develop a method to estimate the steady-state error. It should be mentioned that it is not necessary to actually find the steady-state error. All that is needed is some bounded function that varies with the steady-state error.

When developing a method to estimate the steady-state error there are some characteristics which are desirable. One is that the estimate should remain small while the transients still exist. This is wanted because, since the estimate will be used to vary the rule-base, the estimate should remain small while it is most uncertain (i.e. – while the system is not in steady-state). Another desired characteristic is that past information should be used without using an array to store it. The reason for this is to make the system less sensitive to sudden variations in the response without requiring large amounts of computer memory.

The equation to be used is

$$\overline{sse} = \frac{\left(\overline{E}\right)^3}{\gamma \cdot \overline{\left(V^2\right)} + \left(\overline{E}\right)^2} \tag{3}$$

where $\overline{E} = e + \lambda \cdot \overline{E}_o$

$$\overline{\left(V^2\right)} = \dot{e}^2 + \lambda \cdot \overline{\left(V^2\right)}_o$$

From Eq. (3), it can be seen that as $\gamma \cdot \overline{\left(V^2\right)} \to 0$, $\overline{sse} \to \overline{E}$. Also as $n \to \infty$ and $e = sse$ (actual steady-state error), $\overline{E} \to sse$. Let us see how Eq. (3) has the desired characteristics. The $\gamma \cdot \overline{\left(V^2\right)}$ term in the denominator keeps $\overline{sse}$ small while $\overline{\left(V^2\right)}$ is still large. The term, γ, allows the strength of the effect to be adjusted. The definitions for $\overline{E}$ and $\overline{\left(V^2\right)}$ fulfill the other requirement that past values of the error in position and the

error in velocity be used. This is done by using a weighted running sum of the values instead of just the present value. The variable, λ, is the forgetting factor (weighting variable) which, if between 0 and +1, gives less weight to the past values than the present value.

Stochastic formulations similar to Eq. (3) have been used in the past based on the average error and the standard deviation of the error. Unfortunately, this requires a statistical analysis at each time step. This could become computationally burdensome and require the use of large arrays. Eq. (3) uses the same principle but effectively circumvents these problems.

CHAPTER IV: SIMULATION RESULTS

Using $\overline{sse}$ as the input and Δd (change in d) as the output a simple fuzzy logic controller can be developed for the adaptation routine. The structure of the adaptive fuzzy logic controller is shown in Figure 13.

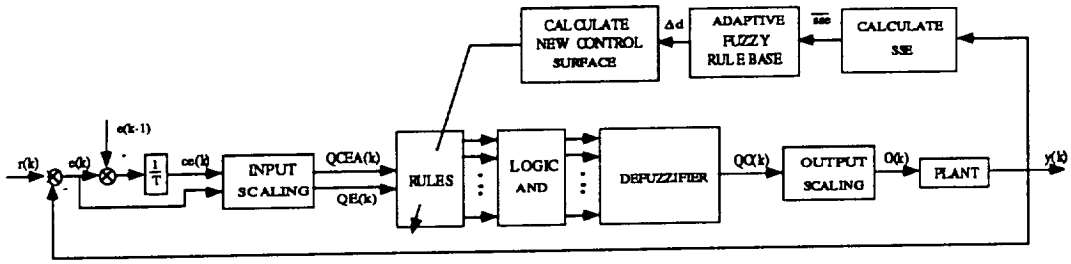


Figure 13: Adaptive fuzzy logic controller block diagram

The results of using this controller on the vertical pendulum already presented is shown in Figure 14.

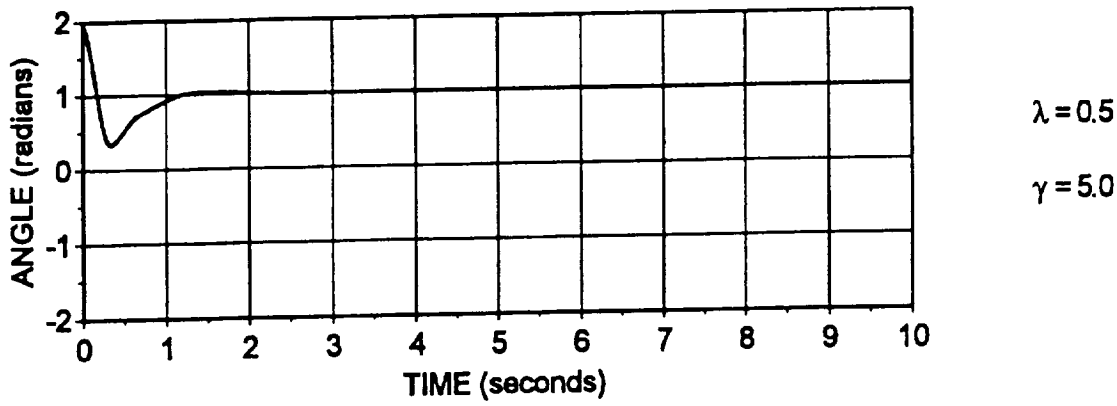


Figure 14: Response of the vertical pendulum using the adaptive fuzzy logic controller

The steady-state error due to the gravitational bias has been eliminated very effectively but it would be interesting to know the effects of varying the λ and γ. These are shown in Figure 15.

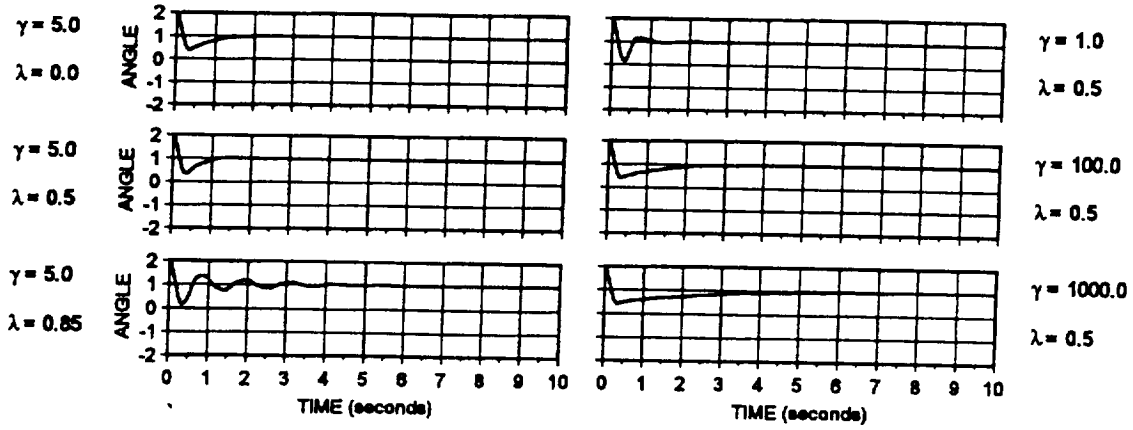


Figure 15: Responses of the vertical pendulum for various λ and γ

The left column of Figure 15 shows a varying λ increasing from top to bottom while the right column shows the same for γ. The results are what should be expected. Since λ is the forgetting factor, increasing it will include more of the past information. This would make the response quicker but less robust. In this example, the system is unstable for $\lambda > 0.85$. Varying γ has a similar affect on the response. Since γ is the weight for the velocity penalty, large values will cause the adaptation routine to wait until more of the transients die out to vary the parameter. Thus, the system will be slower but less likely to be affected by poor estimates. The system in Figure 15 is unstable for $\gamma <$ 1.0. From these findings it should be evident that the controller is made more robust for small λ and large γ. This is important for systems that are complex or have varying parameters.

As an example of a more complex system consider the robot shown in Figure 16(a). This is a drawing of the three degree-of-freedom revolute manipulator currently being designed at the Mars Mission Research Center at North Carolina State University.

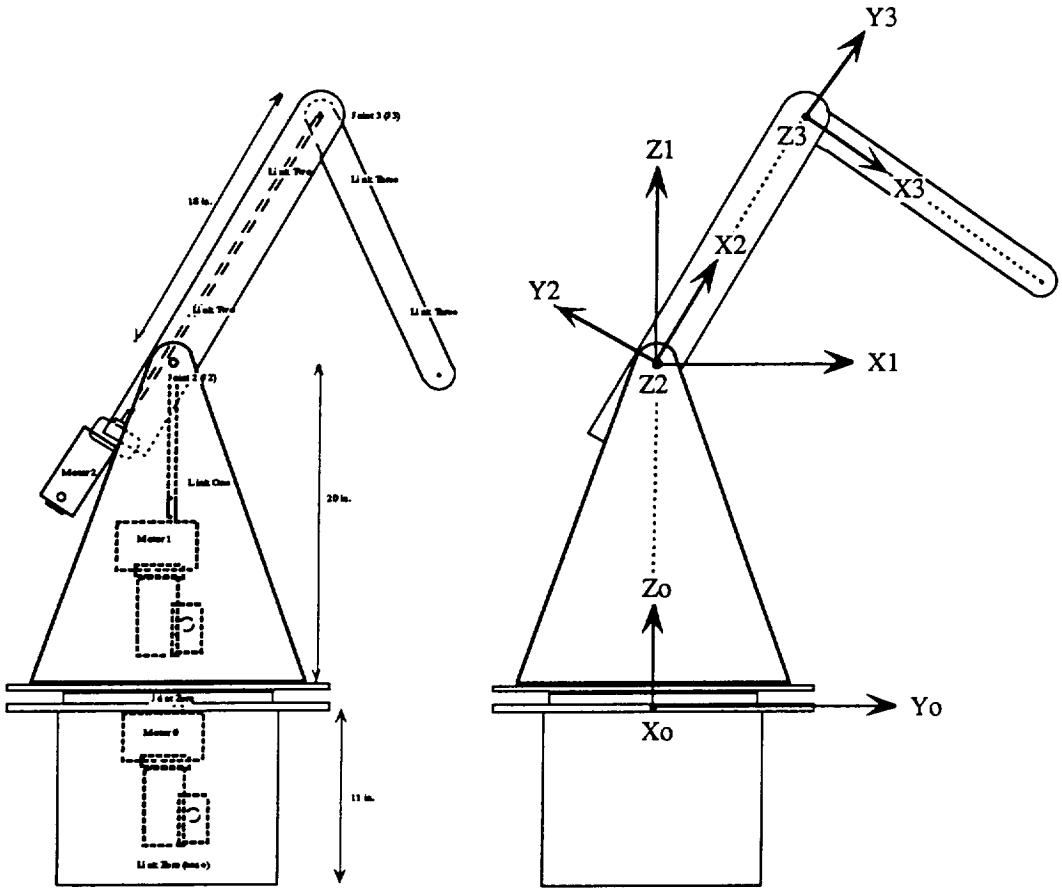


Figure 16: DR106 robot

Figure 16(b) shows how the coordinate system has been defined. The dynamics are as presented in Stanley [12] and can be represented as

$$\ddot{\tilde{\theta}} = \mathbf{M}^{-1}\tilde{\mathrm{R}} + \mathbf{M}^{-1}\tilde{\mathrm{T}}$$

where

$$\mathbf{M} = \begin{bmatrix} 26 + 8C_{23}^2 + 29C_2^2 + 24C_2C_3 & 0 & 0 \\ 0 & 43 + 24C_3 & 8 + 12C_3 \\ 0 & 8 + 12C_3 & 8 \end{bmatrix}$$

$$\tilde{\mathrm{R}} = \begin{Bmatrix} (16C_{23}S_{23} + 24C_2S_{23})\dot{\theta}_1(\dot{\theta}_2 + \dot{\theta}_3) + (24S_2C_{23} + 58S_2C_2)\dot{\theta}_1\dot{\theta}_2 \\ -(8C_{23}S_{23} + 12S_2C_{23} + 29S_2C_2)\dot{\theta}_1^2 + 24S_3\dot{\theta}_2\dot{\theta}_3 + 12S_3\dot{\theta}_3^2 - 20gC_2 - 6gC_{23} \\ -(8C_{23}S_{23} + 12S_2C_{23})\dot{\theta}_1^2 - 12S_3\dot{\theta}_2^2 - 6gC_{23} \end{Bmatrix}$$

$$\tilde{\mathrm{T}} = \begin{Bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{Bmatrix}$$

$$\tilde{\theta} = \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{Bmatrix}$$

$$C_{23} = \cos(\theta_2 + \theta_3)$$

$$S_{23} = \sin(\theta_2 + \theta_3)$$

$$C_2 = \cos(\theta_2)$$

$$C_3 = \cos(\theta_3)$$

$$S_2 = \sin(\theta_2)$$

$$S_3 = \sin(\theta_3)$$

Using these dynamics the adaptive fuzzy logic controller is applied to each joint separately. The coupling is treated as an external disturbance. If each joint starts at 0_{rad} and the desired trajectory is a positive unit step for joints one and two and a negative unit step for joint three a time response like that in Figure 17 occurs. Notice that γ needed to be increased to achieve a good response.

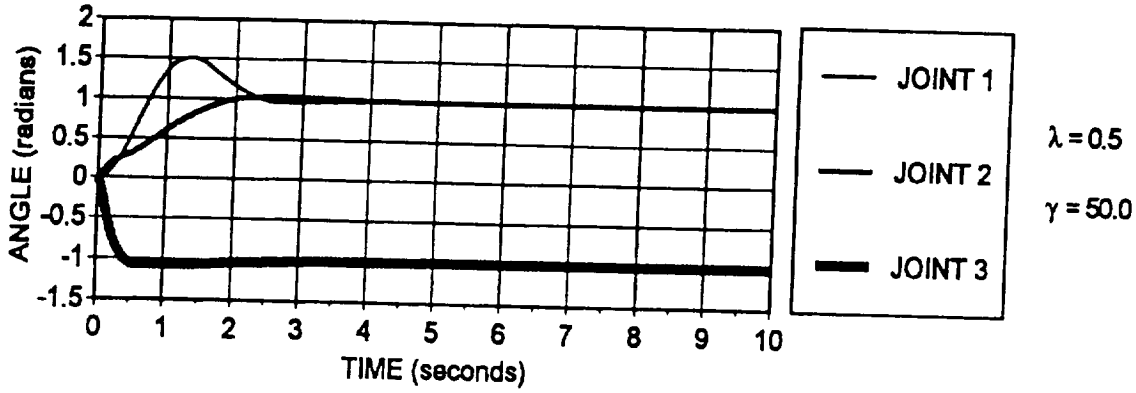


Figure 17: Response of the DR106 using the adaptive fuzzy logic controller

Figure 18 shows three cases where the inertia of the vertical pendulum is not constant. The top graph shows the response for when the inertia changes from 2.0 to 0.5 at 3.0s. This would be similar to robotics applications in which different tools are handled. The middle graph shows the response when the inertia is oscillating between 2.0 and 0.2 at 1Hz. This is the only case where γ had to be varied to get a better response. The last case shown is when the inertia is highly dependent on the angular position. This type of situation could occur in non-symmetrical mechanisms.

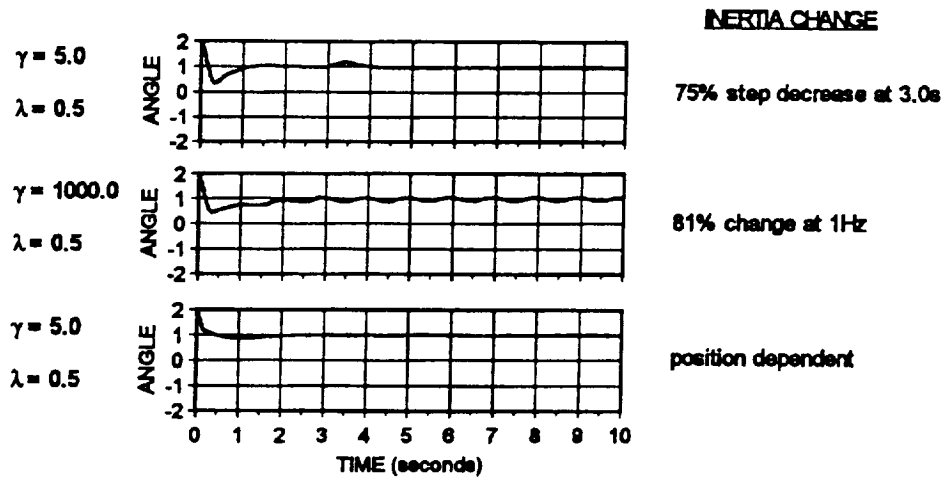


Figure 18: Responses of systems having time varying inertias

One last example of a system having complex characteristics is the vertical pendulum already mentioned but having a time lag between reading the system states and the application of the control force. Since a similar system was tested by Stanley [12] using a non-adaptive fuzzy logic controller, it would be interesting to see if an adaptive fuzzy logic controller is any better. Figure 19 shows the response for several different time lags measured by the number of time steps the application of the control force is delayed.

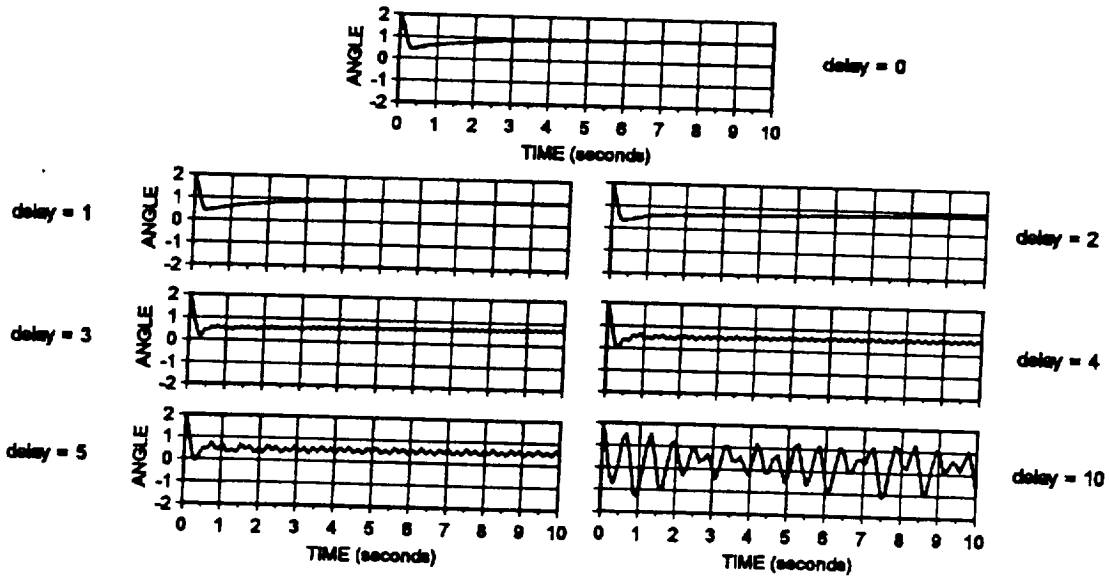


Figure 19: Vertical pendulum having time delays

As can be seen, the system remains stable, although very slowly convergent, until a delay of about ten time steps. When Stanley [12] did a similar test on a horizontal pendulum the system became wildly oscillatory after a delay of three time steps. Therefore, the adaptive fuzzy logic controller does behave better with a system having time delays than a non-adaptive fuzzy logic controller.

CHAPTER V: CONCLUSION AND SUGGESTIONS FOR FUTURE WORK

An adaptive fuzzy logic control algorithm in which the orientation of the defining PD plane is varied on-line by a second fuzzy logic controller has been presented. The adaptation routine used an estimate of the steady-state error having a forgetting factor and a velocity penalty to alter the displacement of the PD plane. The technique was demonstrated on a simple vertical pendulum and gave good results for wide variations in λ and γ. The controller was also demonstrated on more complex systems such as a 3DOF robotic manipulator, various time varying systems and systems with time delays.

The most obvious area in which future research could be done is to develop algorithms which vary the other two parameters defining the PD plane. This would allow the damping rate and frequency to be varied instead of just the steady-state error. The most obvious changes in behavior should occur with systems having time varying parameters and systems having time delays.

Another area open for consideration is using a different type of control surface to define the rule-base. A PD plane was used because it is well known and simple. A higher-order, linear surface could be used giving similar results for more inputs and/or outputs. Another variation would be to use a non-planar surface. An example would be using a surface whose cross-section is like a third-order polynomial to give a sort of dead-zone. Another more complex configuration is a controller having multiple outputs which are coupled. This could be used to either compensate for coupling in the inputs or to introduce coupling into the overall system.

One final suggestion is that different variation methods be investigated. There may very well be a better way to estimate the steady-state error. In fact, the steady-state error may not be the best performance characteristic to measure. A more generalizing change would be to vary the rule-base by making either linear or non-linear coordinate transformations instead of varying the parameters of the control surface. This would allow changes in the shape of the control surface to be made on-line.

The basic concept being investigate in this thesis is the use of a simple mathematical relationship based on a graphical representation to define the rule-base. Thus, parameters in the function defining the rule-base can be varied adaptively which, in turn, changes the entire rule-base in a simple step. The methodology developed in this thesis was done so as to facilitate the presentation of this underlying idea. It should be kept in mind when considering any variations that no attempt was made at optimizing any factors.

CHAPTER VI: LITERATURE CITATIONS

1. Slotine, S. J. and Li, W., "On the Adaptive Control of Robotic Manipulators", *Int. J. of Robotics Research*, Vol. 6, No. 3, pp. 49-59, 1987.

2. Craig, J. J., Hsu, P., and Sastry, S., "Adaptive Control of Mechanical Manipulators", Proc. of the IEEE Int'l Conf. on Robotics and Automation, San Francisco, 1986.

3. Zadeh, L. A., "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes", IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-3, pp. 28-44, 1975.

4. Chang, C. H., "Tuning fuzzy logic controllers via Input and Output Mapping Factors", M.S. Thesis, University of Oklahoma, 1989.

5. King, P. J. and Mamdani, E. H., "The Application of Fuzzy Control Systems to Industrial Processes," Automatica, Vol. 13, pp. 235-242, 1977.

6. Murakami, S. and Maeda, M., "Automobile Speed Control System Using a fuzzy logic controller", Industrial Applications of Fuzzy Control (M. Sugeno, Ed.), pp. 105-123, North Holland, 1985.

7. Mamdani, E.H. and Assilian, S., "An Experiment in Linguistic Synthesis with a fuzzy logic controller", *Int. J. Man-Machine Studies*, Vol. 7, pp. 1-13, 1975.

8. Scharf, E. M., Mandic, N. J., and Mamdani, E. H., "A Self-organizing Algorithm for the Control of a Robot Arm", ISMM Conf. on Mini and Microcomputers and Their Applications, San Antonio, 1983.

9. "Fuzzy Controller Robots and Its Practical Applications (Special Session)", IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems, Raleigh, 1992.

10. Noh, H., Kim, H., Kim, S., Park, M., "Cooperative Mobile Robots Using Fuzzy Algorithm", Proc. of the 1992 IEEE/RSJ International Conf. on Intelligent Robots and Systems, Raleigh, pp. 796-802, 1992.

11. Liu, M.H., "Robotic Deburring Based on Fuzzy Force Control", Proc. of the 1992 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Raleigh, pp. 782-789, 1992.

12. Stanley, R., Gerber, D., Windsor, J., and Lee, G. K. F., "A Fuzzy Controller for Space Manipulator Systems", Conf. on Intelligent Robotic Systems for Space Exploration", Troy, N. Y., 1992.

13. Kouathi, I. and Jones, B., "An Improved Design Procedure for Fuzzy Control Systems", *International Journal of Machine Tools and Manufacture: Design, Research and Application*, Vol. 31, No. 1, 1991, pp. 107-122.

14. Jager, R., Verbuiggen, H. B., Bruÿn, P. M., Krÿgsman, A. J., "Real-Time Fuzzy Expert Control", *IEEE*, Vol. 2, No. 332, 1991, pp. 966-970.

15. Chen, Y. Y. and Jang, J. S., "Imitation of State Feedback Controllers by Fuzzy Linguistic Control Rules", Proc. of the 29th Conf. on Decision and Control, Honolulu, 1990.

16. Tzafestas, S. and Paparrikolopoulos, N. P., "Incremental Fuzzy Expert PID Control", *IEEE Trans. on Industrial Electronics*, Vol. 37, No. 5, Oct. 1990, pp. 365-371.

17. Smith, S. M. and Corner, D. J., "Automated Calibration of a fuzzy logic controller Using a Cell State Space Algorithm", *IEEE Control Systems*, August 1991, pp. 18-28.

18. Acosta, G. G., Mayosky, M. A., Catalfo, J. M., "Fuzzy Logic and Pattern Recognition in Self-Tuning Controller", IEEE/RSJ Int"l Conf. on Intelligent Robots and Systems, Raleigh, 1992.

19. Zhang, B.S., Edmunds, J.M., "On Fuzzy Logic Controllers", IEE Control '91, No. 332, Vol. 2, pp. 961-965, 1991.

20. Kickert, W.J.M., and Mamdami, E.H., *Fuzzy Sets Systems*, Vol. 1, pp. 29-44, 1978.

APPENDICES

An important part of performing good research is ensuring that future researchers can, if so desired, reproduce your results. Towards this end, the three appendices are given. The results presented in this thesis came from many runs of various computer programs. Although they varied in the system model, non-linearities introduced, and the desired results, they have the same basic structure. Because of this a detailed description of one of the programs can serve as a description of all the programs. The program simulating the application of the adaptive fuzzy logic controller to a simple three-link, revolute joint robot, Figure 16, will be the illustrative example. Appendix A gives a detailed progression through the program while Appendix B gives the flow chart for the same program. The actual C code used is listed in Appendix C.

All of the programming was done using Borland C/C++ version 3.1. Since an effort was made to use only conventional commands, the code should work with other C compilers with little or no modification. The structure and methodology were kept very simple so the program should be easy to translate to a different programming language. In fact, several sections were originally programmed in FORTRAN by Stanley [12].

APPENDIX A: EXPLANATION OF PROGRAM

MAIN PROGRAM

I–II In these blocks the output files are opened, the state variables are set to their starting values and other operating variables are set to their initial values.

III The initial values of the adaptation variables are written to an output file. This is done so that the time response of the variables can be plotted later.

IV The **FUZZY LOGIC CONTROL** routine is called to determine the control force given the initial conditions and the initial control surface orientation. This is done for each link separately.

V The initial positions and the control force calculated in IV are written to an output file. This is to facilitate a plotting of their time responses.

VI The **RUNGA-KUTTA** routine is called to determine the new state variables given the old states and control forces.

VII The **STEADY-STATE ERROR APPROXIMATION** routine is called to estimate the steady-state error based on the past response and given the new errors in position and velocity. This is done for each link separately.

VIII The **ADAPTATION FUZZY LOGIC** routine is called to determine the necessary variation in the control surface orientation given the estimated steady-state error. This is done to each link separately.

IX–X Calculate the new adaptation values and write them to an output file.

XI The **FUZZY LOGIC CONTROL** routine is called to determine the control force given the new states and new control surface orientation. This is done for each link separately.

XII Write the new positions and control forces to an output file. Loop back to VI and repeat for NSTEP times.

XIII Upon completing the simulation close all output files.

FUZZY LOGIC CONTROL ROUTINE

I–II Given the link being operated, set E to the appropriate state variable. Using the last value of E and the first two equation on page 12 determine QE and QCEA.

III–IV Determine which membership functions of QE and QCEA will be effective. Since, as mentioned on page 8, no more than four membership functions, for each, will have non-zero values this can greatly reduce the number of rules evaluated and thus speed-up the operation of the routine.

V Set arrays used for holding the membership function values and the consequence block values to zero. These are used so that when LV is performed the correct values will be multiplied.

VI–LIV
Each rule is successively checked for the correct combination of effective membership functions. If the correct combination exists the following blocks are performed:

(a) Each rule corresponds to a unique point in the quantized phase plane determined by the center values of its combination of membership functions.

(b) Determine the degree of membership of QE and QCEA to the fuzzy sets given in the rule. The membership functions used here are a half-cycle of a sine wave. This is the same operation discussed on page 5.

(c) To determine the result of the antecedent block using the logic AND operation given by definition (*i*), page 8 is performed.

(d) Given the coordinates of the rule from (a), the adaptation variables, and the defining equation of the control surface in Eq. (2) determine the value of the control surface at the position of the rule.

LV Determine the quantized control effort using the approximate center of gravity method introduced on page 10. This is simply a quotient in which the numerator is the sum of the products of the result of the logic AND and the value of the control surface for each rule. The denominator is the sum of the results of the logic AND's. Those rules not evaluated in VI-LIV give no contribution due to V.

LVI Write QE and QCEA to an output file so that a phase plot may be drawn.

LVII Scale the result of LV using the method given by the last equation on page 12.

STEADY-STATE ERROR ESTIMATION ROUTINE

I–II The weighted running sums of the position and velocity are updated after determining for which link the routine was called. LAMBDA is also set depending on the link to facilitate varying it independently if necessary.

III Calculate the approximate steady-state error using Eq. (3).

FUZZY LOGIC ADAPTATION ROUTINE

I Set the Max/Min values of the input and output.

II Quantize the incoming approximate steady-state error given the limit from I.

III Determine the membership functions with non-zero values. Unlike the **FUZZY LOGIC CONTROL** routine, which has two inputs, there is a maximum of two rules which will influence the output.

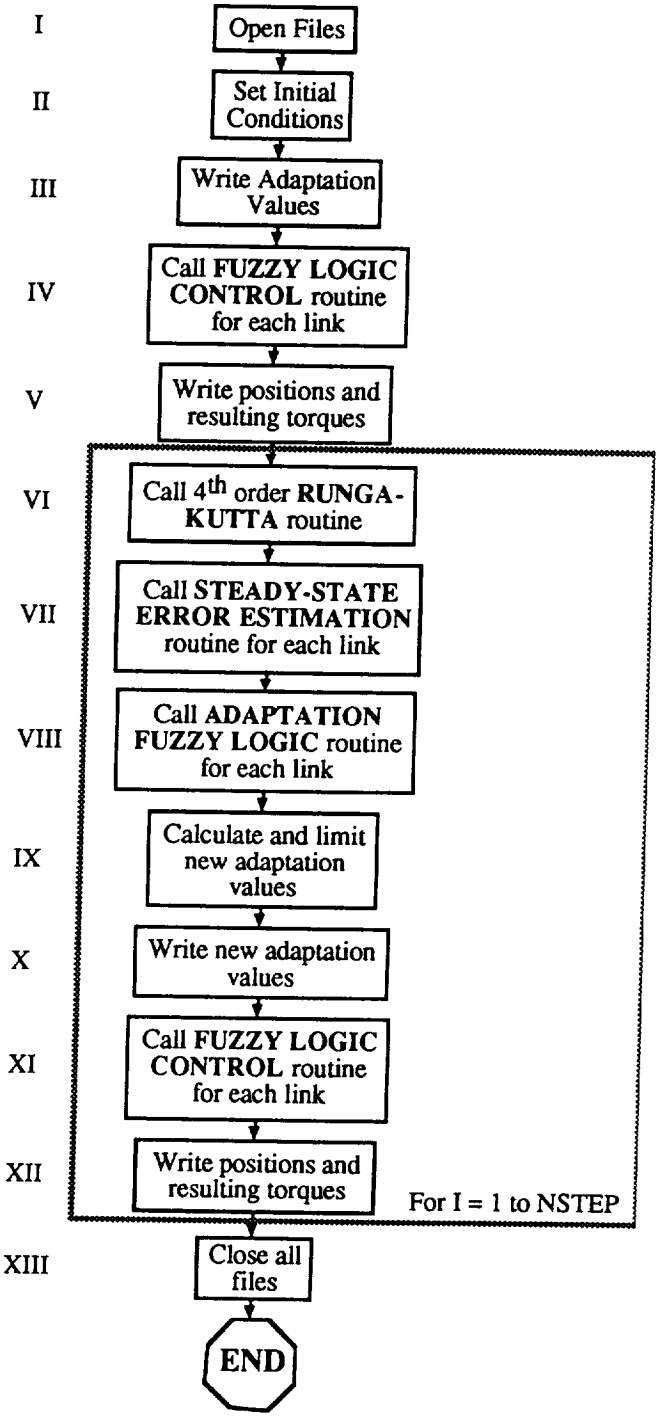
IV Zero the holding arrays as in the **FUZZY LOGIC CONTROL** routine.

V–XI Each rule is successively checked to determine if it is valid. Those that are will be evaluated much the same way as in the **FUZZY LOGIC CONTROL** routine. The major differences, though, are that since there is only one input the logic AND is not used and that the rule-base is fixed.

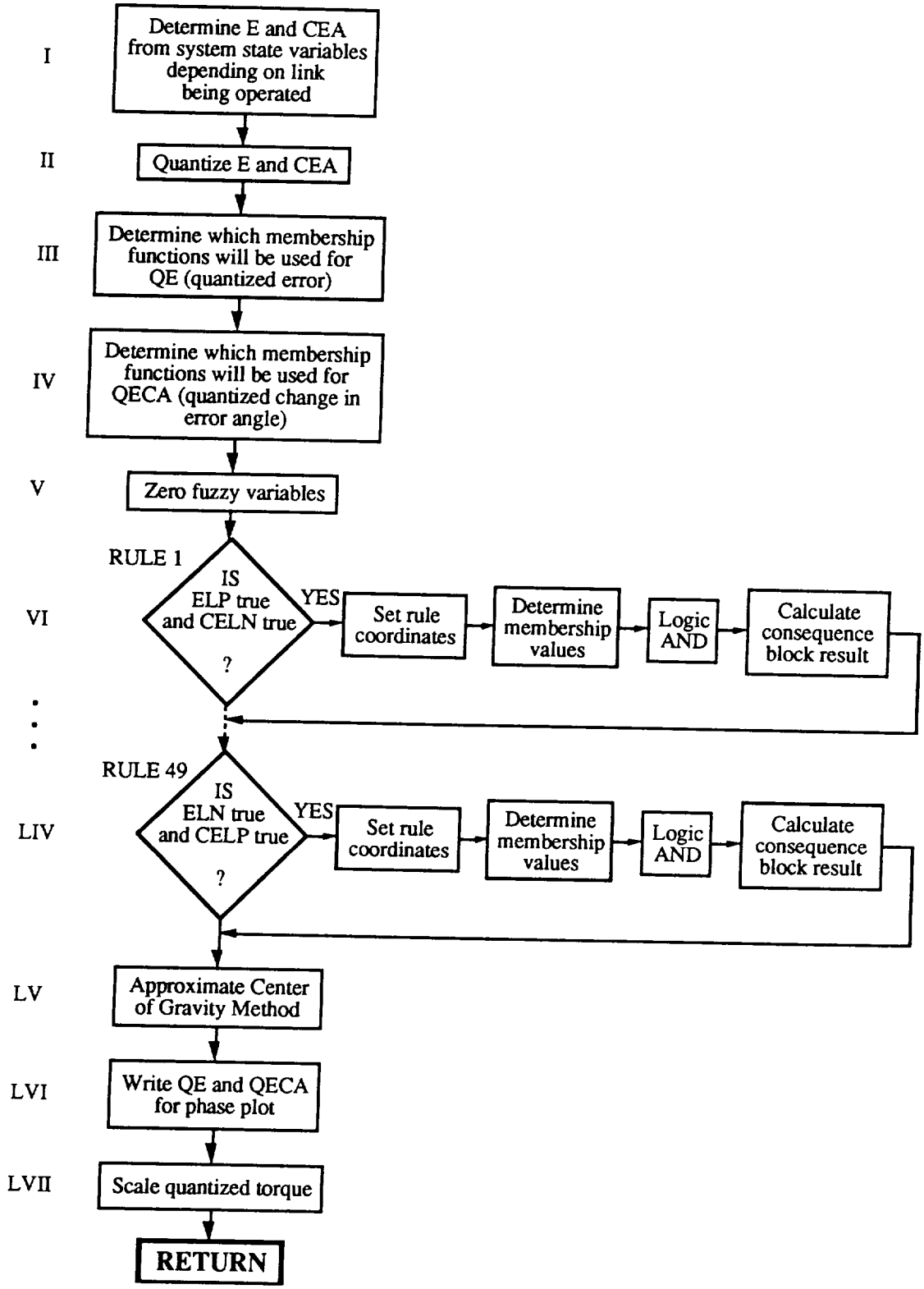
XII-XIII
These blocks are comparable to those in the **FUZZY LOGIC CONTROL** routine.

APPENDIX B: FLOW CHARTS

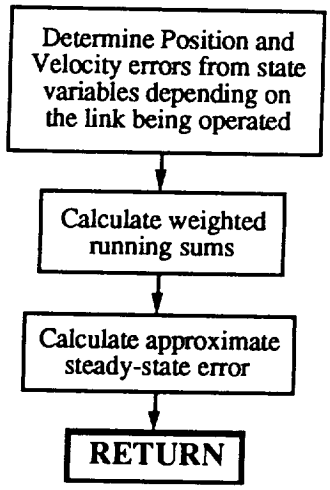
MAIN PROGRAM BLOCK



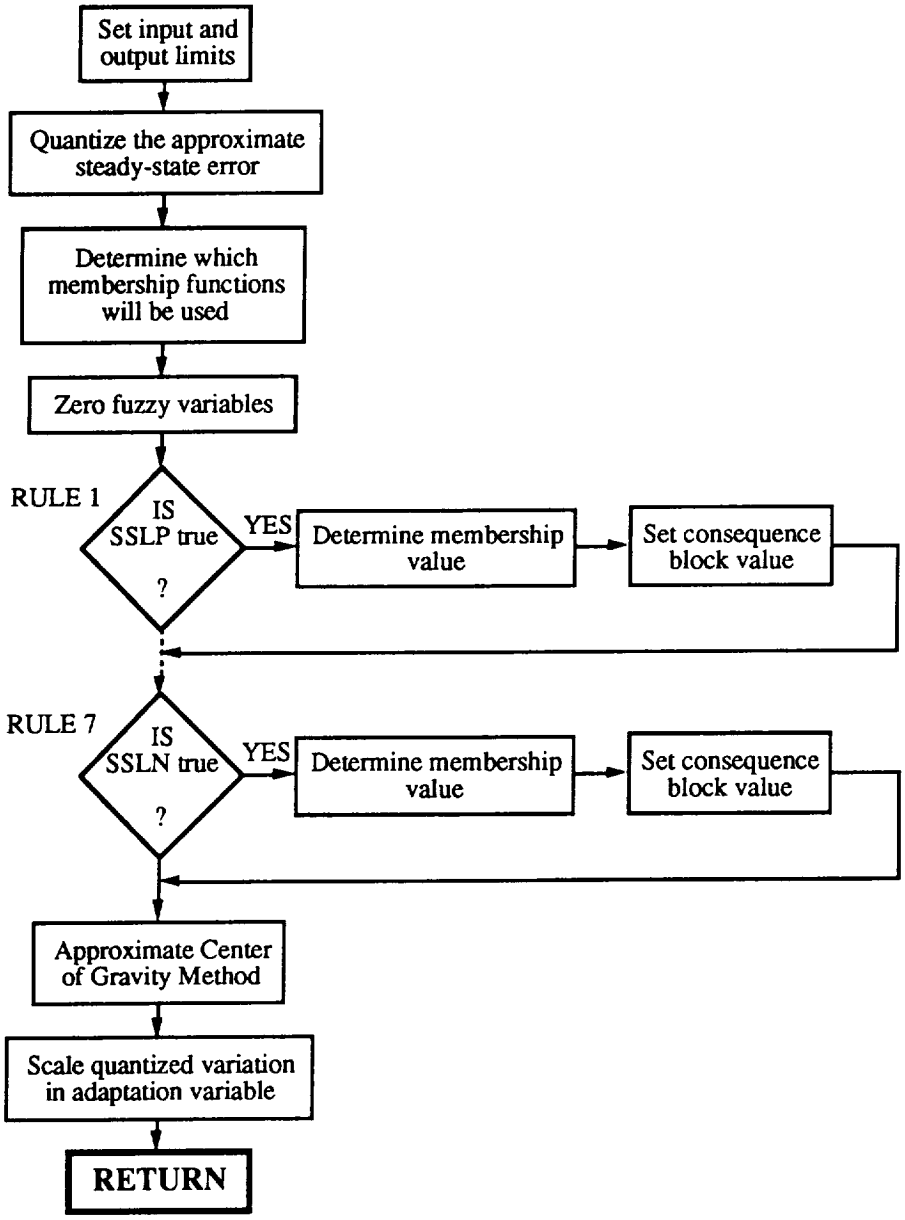
FUZZY LOGIC CONTROL ROUTINE



STEADY-STATE ERROR ESTIMATION ROUTINE



FUZZY LOGIC ADAPTATION ROUTINE



APPENDIX C: PROGRAM LISTING

```
/*
        PURPOSE:  To apply an adaptive Fuzzy-Logic controller to a three-link
                  microbot. (Highly Non-linear Coupled Second Order
                  Differential Equations). The control plane is defined by
                  the displacement along the QE-axis (D), the rotation in the
                  QE-QCEA plane (THETA), and its inclination from the QE-
                  QCEA plane (PHI). D is varied adaptively based upon an
                  approximation of the steady-state error.

        AUTHOR:   Darrell L. Gerber

        DATE:     4/21/93

        VARIABLES: COUNT: Holds the value of the present Runga-Kutta
                          iteration.
                   T#:    The input to link #
                   Td#:   The desired position (Theta desired)for joint #
                   TRIG:  Zero on the first pass and One afterwards
                   NEQ:   Number of state equations
                   NSTEP: Number of times Runga-Kutta subroutine is called
                   DT:    Time interval delta T
                   TIME:  Independent variable
                   X[_]:  Dependent state variables
                   MAX#:  Maximum input allowed in link #
                   D#:    D for link #
                   THETA#: THETA for link #
                   PHI#:  PHI for link #
                   LASTE#: Error for link # during the previous iteration
                   ERROR_SUM#: Weighted running sum of the error for link #
                   VELSQ_ERROR_SUM#: Weighted running sum of the velocity
                          error squared for link #
                   dD:    Change in D as determined by the adaptation routine
                   SSE#:  Approximate Steady-State Error for link #
                   L#:    Link #
*/

#define NEQ 9
#define DT 0.01
#define NSTEP 1000
#define Td1 1.0                 /* Define Desired Positions */
#define Td2 1.0
#define Td3 -1.0
#define PI 3.141592654

#include <stdio.h>
#include <math.h>
#include <mem.h>
#include <stdlib.h>

double LASTE1 = 0.0,LASTE2 = 0.0,LASTE3 = 0.0;
```

```
double ERROR_SUM1 = 0.0,ERROR_SUM2 = 0.0,ERROR_SUM3 = 0.0;
double VELSQ_ERROR_SUM1 = 0.0,VELSQ_ERROR_SUM2 =
0.0,VELSQ_ERROR_SUM3 = 0.0;
int TRIG = 0;


void Right(double [NEQ+1],double *,double *,double *,double *,double *,double
        *,double *,double *,double *,double *,double *);
void State(double [NEQ+1],double [NEQ+1],double *,double *,double *);
void Runga(double [NEQ+1],double *,double *,double *,double *);
void Steady_State_Error(double [NEQ+1],int *,double *);
void Adaptation(double *,double*);
void Fuzzy_Logic(double [NEQ+1],double *,double *,int *,double *,double *,double
        *,FILE *,FILE *,FILE *);


main()
{
   double X[NEQ+1],TIME,T1,T2,T3,MAX1,MAX2,MAX3;
   double D1,D2,D3,THETA1,THETA2,THETA3,PHI1,PHI2,PHI3,SSE1,SSE2,
          SSE3,dD;
   int COUNT,L1=1,L2=2,L3=3;

   FILE *file1,*file2,*file3,*file4,*file5;

   if ((file1=fopen("C:/GERBER/data/fROB1.dat","w"))==NULL) printf("Error opening
        file1\n");
   if ((file2=fopen("C:/GERBER/data/fROB11.dat","w"))==NULL) printf("Error opening
        file2\n");
   if ((file3=fopen("C:/GERBER/data/fROB12.dat","w"))==NULL) printf("Error opening
        file3\n");
   if ((file4=fopen("C:/GERBER/data/fROB13.dat","w"))==NULL) printf("Error opening
        file4\n");
   if ((file5=fopen("C:/GERBER/data/fROB1d.dat","w"))==NULL) printf("Error opening
        file5\n");


   TIME = 0.0;

   X[1] = 0.0;                  /*  Initial Conditions  */
   X[2] = -1.0;
   X[3] = 0.0;
   X[4] = 0.0;
   X[5] = -1.0;
   X[6] = 0.0;
   X[7] = 0.0;
   X[8] = 1.0;
   X[9] = 0.0;

   MAX1 = 500.0;                /*  Set Input Limits  */
   MAX2 = 500.0;
   MAX3 = 150.0;
```

```
D1 = D2 = D3 = 0.0;          /* Initial Conditions of the control planes */
THETA1 = THETA2 = THETA3 = PI/4;
PHI1 = PHI2 = PHI3 = PI/4;

fprintf(file5,"%f %f %f\n",D1,D2,D3);


            /* Determine Control Efforts */

Fuzzy_Logic(X,&T1,&MAX1,&L1,&D1,&THETA1,&PHI1,file2,file3,file4);
Fuzzy_Logic(X,&T2,&MAX2,&L2,&D2,&THETA2,&PHI2,file2,file3,file4);
Fuzzy_Logic(X,&T3,&MAX3,&L3,&D3,&THETA3,&PHI3,file2,file3,file4);
TRIG = 1;

fprintf(file1,"%f %f %f\n",X[2]+Td1,X[5]+Td2,X[8]+Td3);

for(COUNT=1;COUNT<=NSTEP;COUNT++)
{
   Runga(X,&TIME,&T1,&T2,&T3);      /* Calculate New States */
   Steady_State_Error(X,&L1,&SSE1); /* Approximate SSE's    */
   Steady_State_Error(X,&L2,&SSE2);
   Steady_State_Error(X,&L3,&SSE3);
   Adaptation(&SSE1,&dD);           /* Calculate Variations in D's */
   Adaptation(&SSE2,&dD);
   Adaptation(&SSE3,&dD);
   D1 = D1 + dD;                    /* Calculate new D's */
   D2 = D2 + dD;
   D3 = D3 + dD;
   if(D1>=20.0) D1 = 20.0;          /* Limit D's */
   if(D1<=-20.0) D1 = -20.0;
   if(D2>=20.0) D2 = 20.0;
   if(D2<=-20.0) D2 = -20.0;
   if(D3>=20.0) D3 = 20.0;
   if(D3<=-20.0) D3 = -20.0;

   fprintf(file5,"%f %f %f\n",D1,D2,D3);

                        /* Determine Control Efforts */

   Fuzzy_Logic(X,&T1,&MAX1,&L1,&D1,&THETA1,&PHI1,file3,file4,file5);
   Fuzzy_Logic(X,&T2,&MAX2,&L2,&D2,&THETA2,&PHI2,file3,file4,file5);
   Fuzzy_Logic(X,&T3,&MAX3,&L3,&D3,&THETA3,&PHI3,file3,file4,file5);

   fprintf(file1,"%f %f %f\n",X[2]+Td1,X[5]+Td2,X[8]+Td3);
}

fcloseall();

return 0;

}
```

```
/*

                PURPOSE:    To calculate the mass matrix entries and the
                            nonlinear contributions.

                AUTHOR:     Robert J. Stanley II

                DATE:       8/5/92

                TRANSLATION: Darrell L. Gerber

                DATE:       1/13/93

                VARIABLES:  R#:   The nonlinear terms of link #
                            G:    Gravity
                            DET:  The determinant of the mass matrix divided by M11
*/


void Right(double X[NEQ+1],double *R1,double *R2,double *R3,double *M11,double
*M22,double *M33,double *M23, double *DET,double *T1,double *T2,double *T3)
{

  double G,C23,S23,C2,C3,S2;
  double S3,C23S23,C2S23,S2C23,S2C2;

  G = 9.81;
  C23=cos(X[5]+Td2+X[8]+Td3);
  S23=sin(X[5]+Td2+X[8]+Td3);
  C2=cos(X[5]+Td2);
  C3=cos(X[8]+Td3);
  S2=sin(X[5]+Td2);
  S3=sin(X[8]+Td3);
  C23S23=C23*S23;
  C2S23=C2*S23;
  S2C23=S2*C23;
  S2C2=S2*C2;

  *M11=26.0+8.0*C23*C23+29.0*C2*C2+24.0*C2*C23;
  *M22=43.0+24.0*C3;
  *M23=8.0+12.0*C3;
  *M33=8.0;
  *DET=*M22*(*M33)-*M23*(*M23);

*R1=(16.0*C23S23+24.0*C2S23)*X[3]*(X[6]+X[9])+(24.0*S2C23+58.0*S2C2)*X[3]*
     X[6]+*T1;
*R2=-(8.0*C23S23+12.0*S2C23+12.0*C2S23+29.0*S2C2)*X[3]*X[3]+24.0*S3
       *X[6]*X[9]+12.0*S3*X[9]*X[9]-20.0*G*C2-6.0*G*C23+*T2;
*R3=-(8.0*C23S23+12.0*C2S23)*X[3]*X[3]-12.0*S3*X[6]*X[6]-6.0*G*C23+*T3;

}
```

```
/*
                PURPOSE:    To compute the present state of the dynamic system.

                AUTHOR:     Robert J. Stanley II

                DATE:       8/5/92

                TRANSLATION:  Darrell L. Gerber

                DATE:       1/13/93

                VARIABLES:Y:    System states
                          F:    States in Runga-Kutta format

*/

void State (double F[NEQ+1],double Y[NEQ+1],double *T1,double *T2,double *T3)
{

   double R1,R2,R3,M11,M22,M33,M23,DET;

   Right(Y,&R1,&R2,&R3,&M11,&M22,&M33,&M23,&DET,T1,T2,T3);

   F[1]=Y[2];              /*  Define state variables  */
   F[2]=Y[3];
   F[3]=R1/M11;
   F[4]=Y[5];
   F[5]=Y[6];
   F[6]=(R2*M33/DET)-(R3*M23/DET);
   F[7]=Y[8];
   F[8]=Y[9];
   F[9]=-(R2*M23/DET)+(R3*M22/DET);

}
```

```
/*
                PURPOSE:    Use a Fourth-Order Runga-Kutta routine to calculate
                            the next state vector.

                AUTHOR:     Robert J. Stanley II

                DATE:       8/5/92

                TRANSLATION:  Darrell L. Gerber

                DATE:       1/13/93

                VARIABLES:G#:    Variable Gains

*/


void Runga(double X[NEQ+1],double *TIME,double *T1,double *T2,double *T3)
{

   double Y[NEQ+1],F[NEQ+1],G1[NEQ+1],G2[NEQ+1],G3[NEQ+1],G4[NEQ+1];
   int I;

   for(I=1;I<=NEQ;I++) Y[I] = X[I];

   State(F,Y,T1,T2,T3);
   for(I=1;I<=NEQ;I++) G1[I]=DT*F[I];
   *TIME=*TIME+DT/2.0;
   for(I=1;I<=NEQ;I++) Y[I]=X[I]+G1[I]/2.0;
   State(F,Y,T1,T2,T3);
   for(I=1;I<=NEQ;I++)
   {
      G2[I]=DT*F[I];
      Y[I]=X[I]+G2[I]/2.0;
   }
   State(F,Y,T1,T2,T3);
   for(I=1;I<=NEQ;I++)
   {
      G3[I]=DT*F[I];
      Y[I]=X[I]+G3[I];
   }
   *TIME=*TIME+DT/2.0;
   State(F,Y,T1,T2,T3);
   for(I=1;I<=NEQ;I++) G4[I]=DT*F[I];
   for(I=1;I<=NEQ;I++) X[I]=X[I]+(G1[I]+2.0*(G2[I]+G3[I])+G4[I])/6.0;


}
```

```
/*

PURPOSE:  Given a position calculate the torque required to
          drive the error to zero using a Fuzzy-Logic Control
          surface approximating a PD-plane as defined by the
          displacement along the QE-axis (D), the rotation in the
          QE-QCEA plane (THETA), and the inclination from the
          QE-QCEA plane (PHI).

AUTHOR:   Darrell L. Gerber

DATE:     4/21/93

VARIABLES:E:      Error
          CEA:  Change in error angle
          LASTE#: The last error in link #
          PI:   3.14159
          QE:   Quantized value of the error
          QECA: Quantized value of the error change
          u:    Membership function value
          UU:   Universe of discourse value
          NUM:  Numerator of the input value
          DEN:  Denominator of the input value
          Ye:   Temp variable for the error membership function
          Yec:  Temp variable for the change in error
                membership function
          INPUT: The quantized input to the plant
          TORQUE: The actual input to the plant
          N:    The number of rules
          I:    Count variable
          ELP:  Linguistic value Error Large Positive
          EMP:  Linguistic value Error Medium Positive
          ESP:  Linguistic value Error Small Positive
          EZE:  Linguistic value Error Zero
          ESN:  Linguistic value Error Small Negative
          EMN:  Linguistic value Error Medium Negative
          ELN:  Linguistic value Error Large Negative
          CELP: Linguistic value Change in Error Large Positive
          CEMP: Linguistic value Change in Error Medium Positive
          CESP: Linguistic value Change in Error Small Positive
          CEZE: Linguistic value Change in Error Zero
          CESN: Linguistic value Change in Error Small Negative
          CEMN:Linguistic value Change in Error Medium Negative
          CELN: Linguistic value Change in Error Large Negative
          X:    Position of the rule along the QE-axis
          Y:    Position of the rule along the QECA-axis
          XX:   System States
          TOR_MAX:  Max/Min allowed torque
          LINK: Link being maneuvered

*/
```

```
void Fuzzy_Logic(double XX[NEQ+1],double *TORQUE,double *TOR_MAX,int
*LINK,double *D,double *THETA,double *PHI,FILE *file3,FILE *file4,FILE *file5)
{
  double X,Y,E,QE,u[50],UU[50],NUM=0.0,DEN=0.0;
  double Ye,Yec,INPUT,CEA,QECA;
  int N,I;
  int ELP=0,EMP=0,ESP=0,EZE=0,ESN=0,EMN=0,ELN=0;
  int CELP=0,CEMP=0,CESP=0,CEZE=0,CESN=0,CEMN=0,CELN=0;


  N=49;

  switch(*LINK)
  {
    case 1:                    /*  E and CEA if link 1  */
      E=XX[2];
      if(!TRIG) CEA=0.0;
      else CEA=atan2(E-LASTE1,0.01);
      LASTE1=E;
      break;

    case 2:                    /*  E and CEA if link 2  */
      E=XX[5];
      if(!TRIG) CEA=0.0;
      else CEA=atan2(E-LASTE2,0.01);
      LASTE2=E;
      break;

    case 3:                    /*  E and CEA if link 3  */
      E=XX[8];
      if(!TRIG) CEA=0.0;
      else CEA=atan2(E-LASTE3,0.01);
      LASTE3=E;
      break;

    default:
      printf("Error In LINK\n");

  }


  QE=E*(6/(PI/3));             /*  Quantize E and CEA  */
  QECA = CEA*(6/(PI/2.0));

  if(QE>=6.0)                  /*  Determine which linguistic values  */
  {                            /*  are applicable for quantized error  */
    QE = 6.0;
    ELP = 1;
  }
  if((QE>=4.0)&&(QE<6.0)) ELP=EMP=1;
  if((QE>=2.0)&&(QE<4.0)) EMP=ESP=1;
  if((QE>=0.0)&&(QE<2.0)) ESP=EZE=1;
```

```
if((QE>=-2.0)&&(QE<0.0)) EZE=ESN=1;
if((QE>=-4.0)&&(QE<-2.0)) ESN=EMN=1;
if((QE>-6.0)&&(QE<-4.0)) EMN=ELN=1;
if(QE<=-6.0)
{
   QE = -6.0;
   ELN = 1;
}


if(QECA>=6.0)              /*  Determine which linguistic values   */
{                          /*  are applicable for quantized change */
   QECA = 6.0;             /*  in error angle                      */
   CELP = 1;
}
if((QECA>=4.0)&&(QECA<6.0)) CELP=CEMP=1;
if((QECA>=2.0)&&(QECA<4.0)) CEMP=CESP=1;
if((QECA>=0.0)&&(QECA<2.0)) CESP=CEZE=1;
if((QECA>=-2.0)&&(QECA<0.0)) CEZE=CESN=1;
if((QECA>=-4.0)&&(QECA<-2.0)) CESN=CEMN=1;
if((QECA>-6.0)&&(QECA<-4.0)) CEMN=CELN=1;
if(QECA<=-6.0)
{
   QECA = -6.0;
   CELN = 1;
}


for(I=0;I<=N;I++) u[I]=UU[I]=0.0;      /*  Initialize membership function  */
                                       /*  value and universe of discourse */
                                       /*  value                           */


                  /***  Rules for D=0.0, THETA=PHI=PI/4.0  ***/


if(ELP&&CELN)              /*  Rule 1: If ELP and CELN then    */
{                          /*          contribution is ZE      */
   X = 6.0;
   Y = -6.0;
   Ye = sin(PI/4*(QE-4.0));
   Yec = sin(PI/4*(QECA+8.0));
   u[1] = min(Ye,Yec);
   UU[1] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +*D*tan(*PHI)/tan(*THETA);
   if(UU[1]>6.0) UU[1] = 6.0;
   if(UU[1]<-6.0) UU[1] = -6.0;
}

if(ELP&&CEMN)              /*  Rule 2:  If ELP and CEMN then   */
{                          /*           contribution is SN     */
   X = 6.0;
```

```
    Y = -4.0;
    Ye = sin(PI/4*(QE-4.0));
    Yec = sin(PI/4*(QECA+6.0));
    u[2] = min(Ye,Yec);
    UU[2] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y + D*tan(*PHI)/tan(*THETA);
    if(UU[2]>6.0) UU[2] = 6.0;
    if(UU[2]<-6.0) UU[2] = -6.0;
}

if(ELP&&CESN)                    /* Rule 3: If ELP and CESN then   */
{                                /*         contribution is MN     */
    X = 6.0;
    Y = -2.0;
    Ye = sin(PI/4*(QE-4.0));
    Yec = sin(PI/4*(QECA+4.0));
    u[3] = min(Ye,Yec);
    UU[3] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +*D*tan(*PHI)/tan(*THETA);
    if(UU[3]>6.0) UU[3] = 6.0;
    if(UU[3]<-6.0) UU[3] = -6.0;
}

if(ELP&&CEZE)                    /* Rule 4: If ELP and CEZE then   */
{                                /*         contribution is LN     */
    X = 6.0;
    Y = 0.0;
    Ye = sin(PI/4*(QE-4.0));
    Yec = sin(PI/4*(QECA+2.0));
    u[4] = min(Ye,Yec);
    UU[4] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +*D*tan(*PHI)/tan(*THETA);
    if(UU[4]>6.0) UU[4] = 6.0;
    if(UU[4]<-6.0) UU[4] = -6.0;
}

if(ELP&&CESP)                    /* Rule 5: If ELP and CESP then   */
{                                /*         contribution is LN     */
    X = 6.0;
    Y = 2.0;
    Ye = sin(PI/4*(QE-4.0));
    Yec = sin(PI/4*(QECA-0.0));
    u[5] = min(Ye,Yec);
    UU[5] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +*D*tan(*PHI)/tan(*THETA);
    if(UU[5]>6.0) UU[5] = 6.0;
    if(UU[5]<-6.0) UU[5] = -6.0;
}

if(ELP&&CEMP)                    /* Rule 6: If ELP and CEMP then   */
{                                /*         contribution is LN     */
    X = 6.0;
    Y = 4.0;
    Ye = sin(PI/4*(QE-4.0));
    Yec = sin(PI/4*(QECA-2.0));
    u[6] = min(Ye,Yec);
```

```
   UU[6] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +*D*tan(*PHI)/tan(*THETA);
   if(UU[6]>6.0) UU[6] = 6.0;
   if(UU[6]<-6.0) UU[6] = -6.0;
}

if(ELP&&CELP)                    /* Rule 7: If ELP and CELP then    */
{                                /*         contribution is LN      */
   X = 6.0;
   Y = 6.0;
   Ye = sin(PI/4*(QE-4.0));
   Yec = sin(PI/4*(QECA-4.0));
   u[7] = min(Ye,Yec);
   UU[7] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +*D*tan(*PHI)/tan(*THETA);
   if(UU[7]>6.0) UU[7] = 6.0;
   if(UU[7]<-6.0) UU[7] = -6.0;
}

if(EMP&&CELN)                    /* Rule 8: If EMP and CELN then    */
{                                /*         contribution is SP      */
   X = 4.0;
   Y = -6.0;
   Ye = sin(PI/4*(QE-2.0));
   Yec = sin(PI/4*(QECA+8.0));
   u[8] = min(Ye,Yec);
   UU[8] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +*D*tan(*PHI)/tan(*THETA);
   if(UU[8]>6.0) UU[8] = 6.0;
   if(UU[8]<-6.0) UU[8] = -6.0;
}

if(EMP&&CEMN)                    /* Rule 9: If EMP and CEMN then    */
{                                /*         contribution is ZE      */
   X = 4.0;
   Y = -4.0;
   Ye = sin(PI/4*(QE-2.0));
   Yec = sin(PI/4*(QECA+6.0));
   u[9] = min(Ye,Yec);
   UU[9] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +*D*tan(*PHI)/tan(*THETA);
   if(UU[9]>6.0) UU[9] = 6.0;
   if(UU[9]<-6.0) UU[9] = -6.0;
}

if(EMP&&CESN)                    /* Rule 10: If EMP and CESN then   */
{                                /*         contribution is SN      */
   X = 4.0;
   Y = -2.0;
   Ye = sin(PI/4*(QE-2.0));
   Yec = sin(PI/4*(QECA+4.0));
   u[10] = min(Ye,Yec);
   UU[10] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
            *D*tan(*PHI)/tan(*THETA);
   if(UU[10]>6.0) UU[10] = 6.0;
   if(UU[10]<-6.0) UU[10] = -6.0;
```

```
 }

 if(EMP&&CEZE)                    /* Rule 11: If EMP and CEZE then  */
 {                                /*          contribution is MN    */
   X = 4.0;
   Y = 0.0;
   Ye = sin(PI/4*(QE-2.0));
   Yec = sin(PI/4*(QECA+2.0));
   u[11] = min(Ye,Yec);
   UU[11] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
            *D*tan(*PHI)/tan(*THETA);
   if(UU[11]>6.0) UU[11] = 6.0;
   if(UU[11]<-6.0) UU[11] = -6.0;
 }

 if(EMP&&CESP)                    /* Rule 12: If EMP and CESP then  */
 {                                /*          contribution is LN    */
   X = 4.0;
   Y = 2.0;
   Ye = sin(PI/4*(QE-2.0));
   Yec = sin(PI/4*(QECA+0.0));
   u[12] = min(Ye,Yec);
   UU[12] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
            *D*tan(*PHI)/tan(*THETA);
   if(UU[12]>6.0) UU[12] = 6.0;
   if(UU[12]<-6.0) UU[12] = -6.0;
 }

 if(EMP&&CEMP)                    /* Rule 13: If EMP and CEMP then  */
 {                                /*          contribution is LN    */
   X = 4.0;
   Y = 4.0;
   Ye = sin(PI/4*(QE-2.0));
   Yec = sin(PI/4*(QECA-2.0));
   u[13] = min(Ye,Yec);
   UU[13] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
            *D*tan(*PHI)/tan(*THETA);
   if(UU[13]>6.0) UU[13] = 6.0;
   if(UU[13]<-6.0) UU[13] = -6.0;
 }

 if(EMP&&CELP)                    /* Rule 14: If EMP and CELP then  */
 {                                /*          contribution is LN    */
   X = 4.0;
   Y = 6.0;
   Ye = sin(PI/4*(QE-2.0));
   Yec = sin(PI/4*(QECA-4.0));
   u[14] = min(Ye,Yec);
   UU[14] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
            *D*tan(*PHI)/tan(*THETA);
   if(UU[14]>6.0) UU[14] = 6.0;
   if(UU[14]<-6.0) UU[14] = -6.0;
```

```
}

if(ESP&&CELN)                    /* Rule 15: If ESP and CELN then   */
{                                /*              contribution is MP */
   X = 2.0;
   Y = -6.0;
   Ye = sin(PI/4*(QE-0.0));
   Yec = sin(PI/4*(QECA+8.0));
   u[15] = min(Ye,Yec);
   UU[15] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
            *D*tan(*PHI)/tan(*THETA);
   if(UU[15]>6.0) UU[15] = 6.0;
   if(UU[15]<-6.0) UU[15] = -6.0;
}

if(ESP&&CEMN)                    /* Rule 16: If ESP and CEMN then   */
{                                /*              contribution is SP */
   X = 2.0;
   Y = -4.0;
   Ye = sin(PI/4*(QE-0.0));
   Yec = sin(PI/4*(QECA+6.0));
   u[16] = min(Ye,Yec);
   UU[16] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
            *D*tan(*PHI)/tan(*THETA);
   if(UU[16]>6.0) UU[16] = 6.0;
   if(UU[16]<-6.0) UU[16] = -6.0;
}

if(ESP&&CESN)                    /* Rule 17: If ESP and CESN then   */
{                                /*              contribution is ZE */
   X = 2.0;
   Y = -2.0;
   Ye = sin(PI/4*(QE-0.0));
   Yec = sin(PI/4*(QECA+4.0));
   u[17] = min(Ye,Yec);
   UU[17] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
            *D*tan(*PHI)/tan(*THETA);
   if(UU[17]>6.0) UU[17] = 6.0;
   if(UU[17]<-6.0) UU[17] = -6.0;
}

if(ESP&&CEZE)                    /* Rule 18: If ESP and CEZE then   */
{                                /*              contribution is SN */
   X = 2.0;
   Y = 0.0;
   Ye = sin(PI/4*(QE-0.0));
   Yec = sin(PI/4*(QECA+2.0));
   u[18] = min(Ye,Yec);
   UU[18] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
            *D*tan(*PHI)/tan(*THETA);
   if(UU[18]>6.0) UU[18] = 6.0;
   if(UU[18]<-6.0) UU[18] = -6.0;
```

```
}

if(ESP&&CESP)                    /* Rule 19: If ESP and CESP then   */
{                                /*          contribution is MN     */
   X = 2.0;
   Y = 2.0;
   Ye = sin(PI/4*(QE-0.0));
   Yec = sin(PI/4*(QECA+0.0));
   u[19] = min(Ye,Yec);
   UU[19] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
            *D*tan(*PHI)/tan(*THETA);
   if(UU[19]>6.0) UU[19] = 6.0;
   if(UU[19]<-6.0) UU[19] = -6.0;
}


if(ESP&&CEMP)                    /* Rule 20: If ESP and CEMP then   */
{                                /*          contribution is LN     */
   X = 2.0;
   Y = 4.0;
   Ye = sin(PI/4*(QE-0.0));
   Yec = sin(PI/4*(QECA-2.0));
   u[20] = min(Ye,Yec);
   UU[20] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
            *D*tan(*PHI)/tan(*THETA);
   if(UU[20]>6.0) UU[20] = 6.0;
   if(UU[20]<-6.0) UU[20] = -6.0;
}


if(ESP&&CELP)                    /* Rule 21: If ESP and CELP then   */
{                                /*          contribution is LN     */
   X = 2.0;
   Y = 6.0;
   Ye = sin(PI/4*(QE-0.0));
   Yec = sin(PI/4*(QECA-4.0));
   u[21] = min(Ye,Yec);
   UU[21] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
            *D*tan(*PHI)/tan(*THETA);
   if(UU[21]>6.0) UU[21] = 6.0;
   if(UU[21]<-6.0) UU[21] = -6.0;
}

if(EZE&&CELN)                    /* Rule 22: If EZE and CELN then   */
{                                /*          contribution is LP     */
   X = 0.0;
   Y = -6.0;
   Ye = sin(PI/4*(QE+2.0));
   Yec = sin(PI/4*(QECA+8.0));
   u[22] = min(Ye,Yec);
   UU[22] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
            *D*tan(*PHI)/tan(*THETA);
```

```
    if(UU[22]>6.0) UU[22] = 6.0;
    if(UU[22]<-6.0) UU[22] = -6.0;
  }

  if(EZE&&CEMN)                    /* Rule 23: If EZE and CEMN then */
  {                                /*          contribution is MP   */
    X = 0.0;
    Y = -4.0;
    Ye = sin(PI/4*(QE+2.0));
    Yec = sin(PI/4*(QECA+6.0));
    u[23] = min(Ye,Yec);
    UU[23] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
    if(UU[23]>6.0) UU[23] = 6.0;
    if(UU[23]<-6.0) UU[23] = -6.0;
  }

  if(EZE&&CESN)                    /* Rule 24: If EZE and CESN then */
  {                                /*          contribution is SP   */
    X = 0.0;
    Y = -2.0;
    Ye = sin(PI/4*(QE+2.0));
    Yec = sin(PI/4*(QECA+4.0));
    u[24] = min(Ye,Yec);
    UU[24] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
    if(UU[24]>6.0) UU[24] = 6.0;
    if(UU[24]<-6.0) UU[24] = -6.0;
  }

  if(EZE&&CEZE)                    /* Rule 25: If EZE and CEZE then */
  {                                /*          contribution is ZE   */
    X = 0.0;
    Y = 0.0;
    Ye = sin(PI/4*(QE+2.0));
    Yec = sin(PI/4*(QECA+2.0));
    u[25] = min(Ye,Yec);
    UU[25] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
    if(UU[25]>6.0) UU[25] = 6.0;
    if(UU[25]<-6.0) UU[25] = -6.0;
  }

  if(EZE&&CESP)                    /* Rule 26: If EZE and CESP then */
  {                                /*          contribution is SN   */
    X = 0.0;
    Y = 2.0;
    Ye = sin(PI/4*(QE+2.0));
    Yec = sin(PI/4*(QECA+0.0));
    u[26] = min(Ye,Yec);
    UU[26] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
```

```
    if(UU[26]>6.0) UU[26] = 6.0;
    if(UU[26]<-6.0) UU[26] = -6.0;
  }

  if(EZE&&CEMP)                    /* Rule 27: If EZE and CEMN then */
  {                                /*          contribution is MN   */
    X = 0.0;
    Y = 4.0;
    Ye = sin(PI/4*(QE+2.0));
    Yec = sin(PI/4*(QECA-2.0));
    u[27] = min(Ye,Yec);
    UU[27] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
    if(UU[27]>6.0) UU[27] = 6.0;
    if(UU[27]<-6.0) UU[27] = -6.0;
  }

  if(EZE&&CELP)                    /* Rule 28: If EZE and CELP then */
  {                                /*          contribution is LN   */
    X = 0.0;
    Y = 6.0;
    Ye = sin(PI/4*(QE+2.0));
    Yec = sin(PI/4*(QECA-4.0));
    u[28] = min(Ye,Yec);
    UU[28] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
    if(UU[28]>6.0) UU[28] = 6.0;
    if(UU[28]<-6.0) UU[28] = -6.0;
  }

  if(ESN&&CELN)                    /* Rule 29: If ESN and CELN then */
  {                                /*          contribution is LP   */
    X = -2.0;
    Y = -6.0;
    Ye = sin(PI/4*(QE+4.0));
    Yec = sin(PI/4*(QECA+8.0));
    u[29] = min(Ye,Yec);
    UU[29] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
    if(UU[29]>6.0) UU[29] = 6.0;
    if(UU[29]<-6.0) UU[29] = -6.0;
  }

  if(ESN&&CEMN)                    /* Rule 30: If ESN and CEMN then */
  {                                /*          contribution is LP   */
    X = -2.0;
    Y = -4.0;
    Ye = sin(PI/4*(QE+4.0));
    Yec = sin(PI/4*(QECA+6.0));
    u[30] = min(Ye,Yec);
    UU[30] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
```

```
    if(UU[30]>6.0) UU[30] = 6.0;
    if(UU[30]<-6.0) UU[30] = -6.0;
  }

  if(ESN&&CESN)                    /* Rule 31: If ESN and CESN then  */
  {                                /*          contribution is MP    */
    X = -2.0;
    Y = -2.0;
    Ye = sin(PI/4*(QE+4.0));
    Yec = sin(PI/4*(QECA+4.0));
    u[31] = min(Ye,Yec);
    UU[31] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
             *D*tan(*PHI)/tan(*THETA);
    if(UU[31]>6.0) UU[31] = 6.0;
    if(UU[31]<-6.0) UU[31] = -6.0;
  }

  if(ESN&&CEZE)                    /* Rule 32: If ESN and CEZE then  */
  {                                /*          contribution is SP    */
    X = -2.0;
    Y = 0.0;
    Ye = sin(PI/4*(QE+4.0));
    Yec = sin(PI/4*(QECA+2.0));
    u[32] = min(Ye,Yec);
    UU[32] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
             *D*tan(*PHI)/tan(*THETA);
    if(UU[32]>6.0) UU[32] = 6.0;
    if(UU[32]<-6.0) UU[32] = -6.0;
  }

  if(ESN&&CESP)                    /* Rule 33: If ESN and CESP then  */
  {                                /*          contribution is ZE    */
    X = -2.0;
    Y = 2.0;
    Ye = sin(PI/4*(QE+4.0));
    Yec = sin(PI/4*(QECA+0.0));
    u[33] = min(Ye,Yec);
    UU[33] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
             *D*tan(*PHI)/tan(*THETA);
    if(UU[33]>6.0) UU[33] = 6.0;
    if(UU[33]<-6.0) UU[33] = -6.0;
  }

  if(ESN&&CEMP)                    /* Rule 34: If ESN and CEMP then  */
  {                                /*          contribution is SN    */
    X = -2.0;
    Y = 4.0;
    Ye = sin(PI/4*(QE+4.0));
    Yec = sin(PI/4*(QECA-2.0));
    u[34] = min(Ye,Yec);
    UU[34] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
             *D*tan(*PHI)/tan(*THETA);
```

```
    if(UU[34]>6.0) UU[34] = 6.0;
    if(UU[34]<-6.0) UU[34] = -6.0;
  }

  if(ESN&&CELP)                    /* Rule 35: If ESN and CELP then   */
  {                                /*              contribution is MN  */
    X = -2.0;
    Y = 6.0;
    Ye = sin(PI/4*(QE+4.0));
    Yec = sin(PI/4*(QECA-4.0));
    u[35] = min(Ye,Yec);
    UU[35] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
    if(UU[35]>6.0) UU[35] = 6.0;
    if(UU[35]<-6.0) UU[35] = -6.0;
  }

  if(EMN&&CELN)                    /* Rule 36: If EMN and CELN then */
  {                                /*              contribution is LP  */
    X = -4.0;
    Y = -6.0;
    Ye = sin(PI/4*(QE+6.0));
    Yec = sin(PI/4*(QECA+8.0));
    u[36] = min(Ye,Yec);
    UU[36] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
    if(UU[36]>6.0) UU[36] = 6.0;
    if(UU[36]<-6.0) UU[36] = -6.0;
  }

  if(EMN&&CEMN)                    /* Rule 37: If EMN and CEMN then*/
  {                                /*              contribution is LP  */
    X = -4.0;
    Y = -4.0;
    Ye = sin(PI/4*(QE+6.0));
    Yec = sin(PI/4*(QECA+6.0));
    u[37] = min(Ye,Yec);
    UU[37] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
    if(UU[37]>6.0) UU[37] = 6.0;
    if(UU[37]<-6.0) UU[37] = -6.0;
  }

  if(EMN&&CESN)                    /* Rule 38: If EMN and CESN then */
  {                                /*              contribution is LP  */
    X = -4.0;
    Y = -2.0;
    Ye = sin(PI/4*(QE+6.0));
    Yec = sin(PI/4*(QECA+4.0));
    u[38] = min(Ye,Yec);
    UU[38] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
```

```
    if(UU[38]>6.0) UU[38] = 6.0;
    if(UU[38]<-6.0) UU[38] = -6.0;
  }

  if(EMN&&CEZE)                    /* Rule 39: If EMN and CEZE then */
  {                                /*          contribution is MP   */
    X = -4.0;
    Y = 0.0;
    Ye = sin(PI/4*(QE+6.0));
    Yec = sin(PI/4*(QECA+2.0));
    u[39] = min(Ye,Yec);
    UU[39] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
             *D*tan(*PHI)/tan(*THETA);
    if(UU[39]>6.0) UU[39] = 6.0;
    if(UU[39]<-6.0) UU[39] = -6.0;
  }

  if(EMN&&CESP)                    /* Rule 40: If EMN and CESP then */
  {                                /*          contribution is SP   */
    X = -4.0;
    Y = 2.0;
    Ye = sin(PI/4*(QE+6.0));
    Yec = sin(PI/4*(QECA+0.0));
    u[40] = min(Ye,Yec);
    UU[40] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
             *D*tan(*PHI)/tan(*THETA);
    if(UU[40]>6.0) UU[40] = 6.0;
    if(UU[40]<-6.0) UU[40] = -6.0;
  }

  if(EMN&&CEMP)                    /* Rule 41: If EMN and CEMP then*/
  {                                /*          contribution is ZE   */
    X = -4.0;
    Y = 4.0;
    Ye = sin(PI/4*(QE+6.0));
    Yec = sin(PI/4*(QECA-2.0));
    u[41] = min(Ye,Yec);
    UU[41] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
             *D*tan(*PHI)/tan(*THETA);
    if(UU[41]>6.0) UU[41] = 6.0;
    if(UU[41]<-6.0) UU[41] = -6.0;
  }

  if(EMN&&CELP)                    /* Rule 42: If EMN and CELP then */
  {                                /*          contribution is SN   */
    X = -4.0;
    Y = 6.0;
    Ye = sin(PI/4*(QE+6.0));
    Yec = sin(PI/4*(QECA-4.0));
    u[42] = min(Ye,Yec);
    UU[42] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
             *D*tan(*PHI)/tan(*THETA);
```

```
    if(UU[42]>6.0) UU[42] = 6.0;
    if(UU[42]<-6.0) UU[42] = -6.0;
  }

  if(ELN&&CELN)                    /* Rule 43: If ELN and CELN then  */
  {                                /*          contribution is LP    */
    X = -6.0;
    Y = -6.0;
    Ye = sin(PI/4*(QE+8.0));
    Yec = sin(PI/4*(QECA+8.0));
    u[43] = min(Ye,Yec);
    UU[43] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
             *D*tan(*PHI)/tan(*THETA);
    if(UU[43]>6.0) UU[43] = 6.0;
    if(UU[43]<-6.0) UU[43] = -6.0;
  }

  if(ELN&&CEMN)                    /* Rule 44: If ELN and CEMN then  */
  {                                /*          contribution is LP    */
    X = -6.0;
    Y = -4.0;
    Ye = sin(PI/4*(QE+8.0));
    Yec = sin(PI/4*(QECA+6.0));
    u[44] = min(Ye,Yec);
    UU[44] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
             *D*tan(*PHI)/tan(*THETA);
    if(UU[44]>6.0) UU[44] = 6.0;
    if(UU[44]<-6.0) UU[44] = -6.0;
  }

  if(ELN&&CESN)                    /* Rule 45: If ELN and CESN then  */
  {                                /*          contribution is LP    */
    X = -6.0;
    Y = -2.0;
    Ye = sin(PI/4*(QE+8.0));
    Yec = sin(PI/4*(QECA+4.0));
    u[45] = min(Ye,Yec);
    UU[45] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
             *D*tan(*PHI)/tan(*THETA);
    if(UU[45]>6.0) UU[45] = 6.0;
    if(UU[45]<-6.0) UU[45] = -6.0;
  }

  if(ELN&&CEZE)                    /* Rule 46: If ELN and CEZE then  */
  {                                /*          contribution is LP    */
    X = -6.0;
    Y = 0.0;
    Ye = sin(PI/4*(QE+8.0));
    Yec = sin(PI/4*(QECA+2.0));
    u[46] = min(Ye,Yec);
    UU[46] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
             *D*tan(*PHI)/tan(*THETA);
```

```
    if(UU[46]>6.0) UU[46] = 6.0;
    if(UU[46]<-6.0) UU[46] = -6.0;
  }

  if(ELN&&CESP)                    /* Rule 47: If ELN and CESP then   */
  {                                /*              contribution is MP */
    X = -6.0;
    Y = 2.0;
    Ye = sin(PI/4*(QE+8.0));
    Yec = sin(PI/4*(QECA-0.0));
    u[47] = min(Ye,Yec);
    UU[47] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
    if(UU[47]>6.0) UU[47] = 6.0;
    if(UU[47]<-6.0) UU[47] = -6.0;
  }

  if(ELN&&CEMP)                    /* Rule 48: If ELN and CEMP then   */
  {                                /*              contribution is SP */
    X = -6.0;
    Y = 4.0;
    Ye = sin(PI/4*(QE+8.0));
    Yec = sin(PI/4*(QECA-2.0));
    u[48] = min(Ye,Yec);
    UU[48] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
    if(UU[48]>6.0) UU[48] = 6.0;
    if(UU[48]<-6.0) UU[48] = -6.0;
  }

  if(ELN&&CELP)                    /* Rule 49: If ELN and CELP then   */
  {                                /*              contribution is ZE */
    X = -6.0;
    Y = 6.0;
    Ye = sin(PI/4*(QE+8.0));
    Yec = sin(PI/4*(QECA-4.0));
    u[49] = min(Ye,Yec);
    UU[49] = -tan(*PHI)*X/tan(*THETA) - tan(*PHI)*Y +
              *D*tan(*PHI)/tan(*THETA);
    if(UU[49]>6.0) UU[49] = 6.0;
    if(UU[49]<-6.0) UU[49] = -6.0;
  }

  for(I=1;I<=N;I++)                /* Quantized input using approximate  */
  {                                /* center of gravity method           */
    NUM = NUM+u[I]*UU[I];
    DEN = DEN+u[I];
  }

  if((DEN<=0.0001)&&(DEN>=0.0)) DEN = 0.0001; /* Division by zero protection */
  if((DEN>=-0.0001)&&(DEN<0.0)) DEN = -0.0001;
```

```
    INPUT = NUM/DEN;                                /* Scaled input */

    if(*LINK==1) fprintf(file3,"%f %f\n",QE,QECA);
    if(*LINK==2) fprintf(file4,"%f %f\n",QE,QECA);
    if(*LINK==3) fprintf(file5,"%f %f\n",QE,QECA);

    *TORQUE = INPUT*(*TOR_MAX/6.0);          /* Actual input */

  }



  /*

                PURPOSE:    To approximate the steady-state error of the system by
                            EBAR^3/(VSQBAR + EBAR^2)
                      where
                            EBAR = E + ESUM*LAMBDA
                            VSQBAR = V^2 + VELSQSUM*LAMBDA
                            LAMBDA = forgetting factor
                            ESUM = previous EBAR
                            VSQSUM = previous VSQBAR
                            E = error
                            V = velocity error

                AUTHOR:     Darrell L. Gerber

                DATE:       4/21/93

                VARIABLES: X:    System states
                       LINK: Link being estimated
                       SSE:    Steady-state error estimate for LINK
                       ERROR: Current error in position
                       VEL_ERROR:  Current error in velocity
                       ERROR_BAR:  EBAR
                       VEL_ERROR_BAR:  VSQBAR

  */

  void Steady_State_Error(double X[NEQ+1],int *LINK,double *SSE)
  {
    double ERROR,VEL_ERROR,LAMBDA,ERROR_BAR,VEL_ERROR_BAR;

    switch(*LINK)
    {
      case 1:
        ERROR = X[2];
        VEL_ERROR = X[3];
        LAMBDA = 0.5;
```

```
        ERROR_SUM1 = LAMBDA*ERROR_SUM1 + ERROR;
        VELSQ_ERROR_SUM1 = LAMBDA*VELSQ_ERROR_SUM1 +
              VEL_ERROR*VEL_ERROR;
        ERROR_BAR = ERROR_SUM1;
        VEL_ERROR_BAR = VELSQ_ERROR_SUM1;
        break;

      case 2:
        ERROR = X[5];
        VEL_ERROR = X[6];
        LAMBDA = 0.5;
        ERROR_SUM2 = LAMBDA*ERROR_SUM2 + ERROR;
        VELSQ_ERROR_SUM2 = LAMBDA*VELSQ_ERROR_SUM2 +
              VEL_ERROR*VEL_ERROR;
        ERROR_BAR = ERROR_SUM2;
        VEL_ERROR_BAR = VELSQ_ERROR_SUM2;
        break;

      case 3:
        ERROR = X[8];
        VEL_ERROR = X[9];
        LAMBDA = .5;
        ERROR_SUM3 = LAMBDA*ERROR_SUM3 + ERROR;
        VELSQ_ERROR_SUM3 = LAMBDA*VELSQ_ERROR_SUM3 +
              VEL_ERROR*VEL_ERROR;
        ERROR_BAR = ERROR_SUM3;
        VEL_ERROR_BAR = VELSQ_ERROR_SUM3;
        break;

      default:
        printf("Error In LINK \n");

    }

    *SSE = ERROR_BAR*ERROR_BAR*ERROR_BAR/(50.0*VEL_ERROR_BAR +
              ERROR_BAR*ERROR_BAR);

}


/*

              PURPOSE:    To use a SISO Fuzzy-Logic controller to determine
                          the necssary change in D based on the approximated
                          steady_state error.

              AUTHOR:     Darrell L. Gerber

              DATE:       4/21/93

              VARIABLES:  SSE_RANGE:  Maximum expected steady-state error
```

```
                        MAX_dD: Maximum allowed change in D
                        QSSE:   Quantized steady-state error
                        SSLP:   Steady-state error Large Positive
                        SSMP:   Steady-state error Medium Positive
                        SSSP:   Steady-state error Small Positive
                        SSZE:   Steady-state error Zero
                        SSSN:   Steady-state error Small Negative
                        SSMN:   Steady-state error Medium Negative
                        SSLN:   Steady-state error Large Negative
                        u:      Membership function value
                        UU:     Universe of discourse value
                        NUM:    Numerator of the input value
                        DEN:    Denominator of the input value
                        Ye:     Temp variable for the error membership
                                function
                        Yec:    Temp variable for the change in error
                                membership function
                        INPUT:  The quantized change in D

*/

void Adaptation(double *SSE,double *dD)
{
  double SSE_RANGE,QSSE,u[8],UU[8],NUM=0.0,DEN=0.0,INPUT,MAX_dD;
  int I,SSLP=0,SSMP=0,SSSP=0,SSZE=0,SSSN=0,SSMN=0,SSLN=0;


  SSE_RANGE = 5.0;          /* Set input/output ranges */
  MAX_dD = 1.0;

  QSSE = *SSE*6/SSE_RANGE;  /* Quantize approximate steady-state error */

  if(QSSE>=6.0)             /* Determine applicable membership functions */
  {
    QSSE = 6.0;
    SSLP = 1;
  }
  if((QSSE>=4.0)&&(QSSE<6.0)) SSLP=SSMP=1;
  if((QSSE>=2.0)&&(QSSE<4.0)) SSMP=SSSP=1;
  if((QSSE>=0.0)&&(QSSE<2.0)) SSSP=SSZE=1;
  if((QSSE>=-2.0)&&(QSSE<0.0)) SSZE=SSSN=1;
  if((QSSE>=-4.0)&&(QSSE<-2.0)) SSSN=SSMN=1;
  if((QSSE>=-6.0)&&(QSSE<-4.0)) SSMN=SSLN=1;
  if(QSSE<-6.0)
  {
    QSSE=-6.0;
    SSLN=1;
  }
```

```
for(I=1;I<=7;I++) u[I]=UU[I]=0.0;        /* Initialize membership funtction value */
                                         /* and Universe of Discourse value       */

if(SSLP)                                 /* Rule 1: If QSSE is LP then LN */
{
   u[1] = sin(PI/4.0*(QSSE-4.0));
   UU[1] = -6.0;
}

if(SSMP)                                 /* Rule 2: If QSSE is MP then MN */
{
   u[2] = sin(PI/4.0*(QSSE-2.0));
   UU[2] = -4.0;
}

if(SSSP)                                 /* Rule 3: If QSSE is SP then SN */
{
   u[3] = sin(PI/4.0*(QSSE-0.0));
   UU[3] = -2.0;
}

if(SSZE)                                 /* Rule 4: If QSSE is ZE then ZE */
{
   u[4] = sin(PI/4.0*(QSSE+2.0));
   UU[4] = 0.0;
}

if(SSSN)                                 /* Rule 5: If QSSE is SN then SP */
{
   u[5] = sin(PI/4.0*(QSSE+4.0));
   UU[5] = 2.0;
}

if(SSMN)                                 /* Rule 6: If QSSE is MN then MP */
{
   u[6] = sin(PI/4.0*(QSSE+6.0));
   UU[6] = 4.0;
}

if(SSLN)                                 /* Rule 7: If QSSE is LN then LP */
{
   u[7] = sin(PI/4.0*(QSSE+8.0));
   UU[7] = 6.0;
}


for(I=1;I<=7;I++)                        /* Approximate Center of Gravity Method */
{
   NUM = NUM + u[I]*UU[I];
   DEN = DEN + u[I];
}
```

```
    if((DEN<=0.0001)&&(DEN>=0.0)) DEN = 0.0001; /*  Division by zero protection  */
    if((DEN>=-0.0001)&&(DEN<0.0)) DEN = -0.0001;

    INPUT = NUM/DEN;                    /*  Quantized change in D  */

    *dD = INPUT*MAX_dD/6.0;             /*  Actual variation in D  */

}
```